内容 SCILAB 软件光盘

Introduction to Scientific Computing Software— SCILAB

科学计算自由软件

SCILAB 教程

韶自领 赵 曼 陈惠珍 编著

清华大学出版社

内容简介

- 本书是一本介绍功能与MATLAB类似的科学计算自由软件SCILAB的中文书籍。
- 科学计算（如算术、微积分、逻辑推理、数据可视化等）已经成为计算机应用中的主要内容之一。
- 科学计算软件将成为理工科大学，以至高中教育中计算机应用的必备工具。每一个学生都应该掌握它。
- 科学计算软件为培养学生分析问题和解决问题的能力提供了一个更为广阔的空间。
- “开放源码”方式的自由软件使计算机用户可以真正地学习并掌握软件编程中的核心技术。
- SCILAB的用户可以包括：科研人员、工程技术人员、教师、大学生以及高中生。
- 本书配有SCILAB软件光盘（带有Linux，UNIX以及Windows上的应用源程序与执行码）。安装方便，使用简单。
- “创开源，我自豪”。开放、自豪的学习与创造方式将引导你成为真正的计算机专家。

ISBN 7-900643-48-6
9 787900 643483 >
定价：19.00元（含光盘）

科学计算自由软件

——SCILAB 教程

胡包钢　赵　星　康孟珍　编著

清 华 大 学 出 版 社

内 容 简 介

科学计算已经成为计算机应用中的主要内容之一。科学计算软件为培养学生分析问题和解决问题的能力提供了一个更有力的工具和方便的平台。

不同于 MATLAB,以"开放源码"为特征的 SCILAB 可使您真正掌握并发展科学计算软件本身及其核心技术,更重要的是,作者能够拥有利用 SCILAB 编写作品的自主版权。本书是第一本介绍科学计算自由软件 SCILAB 的中文教程,书中的数百个例题将对读者理解和掌握 SCILAB 的各项功能有很大帮助。

SCILAB 的用户包括:科研人员、工程技术人员、教师、大学生和高中生。本书配有 SCILAB 软件光盘(带有 Linux, UNIX 以及 Windows 上的应用源程序与执行码)。该软件安装方便,使用简单。

书　　名:科学计算自由软件——SCILAB 教程

作　　者:胡包钢　赵　星　康孟珍　编著

出 版 者:清华大学出版社(北京清华大学学研大厦,邮编 100084)

http://www.tup.tsinghua.edu.cn

责任编辑:马幸兆

版式设计:韩爱君

印 刷 者:北京鑫丰华彩印有限公司

发 行 者:新华书店总店北京发行所

开　　本:850×1168 1/32　**印张**:9　**字数**:226 千字

版　　次:2003 年 1 月第 1 版　2003 年 1 月第 1 次印刷

书　　号:ISBN 7-900643-48-6

印　　数:0001～5000

定　　价:19.00 元(含光盘)

序一

中国科学院自动化所和法国国立信息与自动化研究院(INRIA)联合创办的中法信息、自动化与应用数学实验室(LIAMA)成立已5年。作为一个开放实验室,LIAMA吸引了中法许多大学与科研单位的科学家,这些科学家紧密合作,从事了许多基础性、公益性的前沿科学研究。5年来,双方研究人员在同一实验室从事3个月以上共同研究的达35人次,成为中法科研合作的典范。最近,双方科学家又在推动发展和普及科学计算自由软件SCILAB。他们组织了基于SCILAB平台的应用软件比赛,又出版本书,以便让更多的同仁们了解和掌握SCILAB。中法科学家还在策划进一步完善这一具有巨大应用前景(尤其是对中国科技与教育界)的开放源代码自由软件,体现了中法科学家共有的社会责任。作为中法实验室的发起人与前任中方主任,我感到十分欣喜。

科学计算软件应是高中以上,尤其是理工科大学以上学生必备的软件,也是其他许多科学研究以及工程应用研究与开发中必备的基础软件平台。与其他一些基础软件平台类似,一些先以自由软件形式出现,后又以商业软件垄断市场的软件,以其昂贵的价格,使人们望而却步。尤其对发展中国家的人们来说,这种现象成为信息时代巨大数字鸿沟的重要原因。这里,我不想对自由软件与商业软件的功过做一般的评论,有关的争论已在全球范围内热

烈地进行，本书的作者在第 10 章也对此发表了精辟的见解，结合本书的发表，我只想说，**对于基础科学计算软件，开放源代码的自由软件应该起主导作用**，虽然现实远远不是如此。

“科学是国际的”、“科学是人类同创、共享的事业”。这是绝大多数有社会责任感的科学家的共识。科学研究，尤其是基础科学研究和涉及教育的基础科学研究绝大多数都是靠各国政府资助的（更准确地说，是靠各国纳税人共同资助的）。像科学计算这样的软件，刚开始都是在大学或公共研究所发展起来的，是经过学术界公开讨论，共同研究与完善中逐步形成的。科学计算方法及其软件是人类智慧的共同结晶，理应成为人类教育与培养下一代的共同财富，成为世界各国科学家进一步从事科学研究的共同工具。然而在商业运作下，一些基础科学计算软件却成了只有少数人能用得起的软件。动辄数千元、甚至数万元价格的软件把世界上绝大多数急需使用该软件的学生、教师、科学工作者排除在外，不仅造成大量的重复劳动和资金的浪费，还导致了事实上难以制止的软件盗版。据商业软件联盟（business software alliance）2001 年的统计，即使在最富有的美国，软件盗版率也达 24%，日本、法国、意大利分别达 37%，40%和 46%。而发展中国家则都高达 60%以上。软件盗版搅乱了真正应商业化的软件市场，还动摇了人们本应具有的最基本的一些道德理念。消除数字鸿沟是当前各国政府普遍关心的问题。信息与通信技术的发展，产生了诸如远程教育、远程医疗、电子政务、电子商务等技术。人们期望，这些技术给发展中国家和被边缘化的贫穷落后地区改善教育和医疗条件，提高政府和企业工作效率带来巨大的好处。然而，高昂的信息产品价格壁垒和对部分产品的垄断是日益加大的数字鸿沟的重要原因。看看下面的统计数据吧。

全世界 50%的计算机在美国，也就是说，美国一国拥有的计算机数是全世界所有其他国家拥有计算机数的总和。东京

一个城市的电话机总数超过了非洲大陆的电话机数的总和。全球因特网(Internet)用户为 5.4 亿左右(2000 年),其中,欧洲、北美洲加上亚洲的日本占 73%.如果只计算美、英、法等 7 大经济强国,则占到 60%。与 Internet 相连的服务器数集中度更为明显。全球服务器数为 9459 万台左右(2000 年),其中,欧洲、北美洲加上亚洲的日本占了 91%,而美、英、法等 7 大经济强国就已占到 79%。在我国,地区之间的差距也令人堪忧。据 2000 年的统计,按每万人因特网用户数比较,Internet 使用率全国最高的是北京(每万人中也仅有 2017 人),其使用率是西藏的近 100 倍,是贵州、河南等省的近 40 倍,甚至是与它毗邻的河北省的近 30 倍。全世界范围内日益扩大的数字鸿沟,在经济全球化的背景下,信息产品被少数跨国公司垄断的日益严重的现状,引起了各国政府、科学工作者、非政府组织以及包括企业界有识之士的关切。

有人说,像基础科学计算这样的本来应成为人类的共同财富,所有人都可自由使用的软件,之所以成为高价格壁垒后的商业软件,是因为自由软件或开放源代码方式的软件开发缺乏严密的组织,科学家们或软件开发人员在无商业利益的驱动下,是不可能将其最终开发成灵活、好用、可靠的软件的,许多科学家也只对软件开发初期的计算原理与计算方法的研究感兴趣,而不愿意把时间花在方法的最终集成与完善上。我认为,这种观点有一定道理。事实上,许多同样模式开发的软件都还没达到类似商业软件的水平,就说明了这一点。因此我并不认为自由软件或开放源代码方式的软件开发是普遍适用的方法,所谓自由方式的"开放源码"与商业方式的"封闭源码"形成的对垒、竞争甚至协同发展,恐怕会持续很长时间。但是,对基础科学计算这样的软件,本应该被广大学生、教师、科学工作者广泛使用,却被高价格壁垒阻止,在这样的形势下,有社会责任感的科学工作

者都应行动起来，义不容辞地承担起开发自由软件的责任。事实上许多科学工作者，包括发达国家和发展中国家的科学工作者已经这样做了，他们应该受到社会的尊重和政府的支持。

SCILAB 是相当于 MATLAB 与 MAPLE 的科学计算基础软件平台，十多年前我在法国的 INRIA 中学习时就有人在开发了。十多年来，INRIA 与法国国立桥梁学院(ENPC)的科学工作者坚持 SCILAB 的开放源代码与自由软件原则，最近又与中法联合实验室的同仁们共同努力准备将其在中国推广普及，这一行为理所当然地受到了中法两国科学家的支持。许多中国高校的学生还积极参与了基于 SCILAB 软件平台的应用软件比赛。中国科技部“863”计划和法国驻华使馆，也对此给予了积极的支持。这本《科学计算自由软件——SCILAB 教程》的出版，是推动该工作在中国进一步开展的重要一步。SCILAB 的“开放源码”特征将会吸引中国研究人员加入到与世界软件同行合作开发的行列中。SCILAB 软件同时是科学计算与教育的平台，它涉及自然科学的众多领域，对基于 SCILAB 软件平台应用软件的合作开发还可为中外科学工作者与教育工作者的合作创造更为广阔的前景。

正如中法实验室(LIAMA)中方主任胡包钢博士所说，在科学、教育这样的公共事业发展中，“开放、同创、共享”的发展模式是值得提倡的。为了人类美好的未来，我们需要携起手来，共同创造和分享科学的成果。

马颂德博士

中国科学技术部副部长

2002 年 5 月于北京

序二

科学计算软件曾在工程界的研究和发展活动中掀起了一场革命。如今，这些软件已经被广泛应用到工业工程实验室中，成为各科研领域中众多学者、教师和学生的必备工具。法国国立信息与自动化研究院(INRIA)无疑是这场革命的先驱者。从1994年开始，它推出的开放源码软件SCILAB积极地推动了这场革命。SCILAB综合了多方面的研究成果，是众多学者心血的结晶，然而SCILAB的诞生更应该归功于它的6位创始者——SCILAB小组成员：INRIA的Francois Delebecque，Claude Comez，Maurice Goursat，Ramine Nikoukhah，Serge Steer以及法国国立桥梁学院(ENPC)的Jean-Phillippe Chancelier。

现在，SCILAB软件及其工具箱已经取得了很大的成功，每月都有来自全球的近万人次登录SCILAB网站，并下载该软件。如此国际化的成功一方面因为该软件本身的句法和基本功能完全可以和行业参照软件MATLAB相媲美，另一方面因为它是完全免费的。此外，这份成功也应归功于其源代码开放的特征。使用者可以完全控制其开发计划，并可通过嵌入最新最先进的技术优化SCILAB软件。不过，成功是没有终点的，对于SCILAB软件完全能够而且应该看得更高，走得更远。我们的目标是：在未来的几年中使SCILAB软件成为一个优秀的科学计算工具；并在教育、研

究以及工业领域内得到认可，成为一个具有国际水平的参照工具；借助于免费的优势促进其发展，加强使用 SCILAB 使用者和参与群体的安全性；确保工业需要和行业先进技术的优先联系。为朝此目标发展，INRIA 决定在 2002 年加大推广使用 SCILAB 软件的力度，并与学术界和工业界的合作者共同建立“SCILAB 共同体”(consortium SCILAB)。

如此宏大的计划如果仅仅局限于法国或是欧洲范围内是远远不够的，只有将其放在国际环境下才能取得成功。因此 SCILAB 小组的成员正努力扩大 SCILAB 软件在国际上的影响。值得一提的是“SCILAB 中国经历”的开始，法国国际农业研究发展中心(CIRAD)的研究员 Philippe de Reffye 功劳不可埋没，在中国的 3 年中，他在中国科学院与 INRIA 共同创建的中法实验室(LIAMA)工作，在那里他用 SCILAB 软件成功地开展了植物建模方面的研究。对于此项应用 SCILAB 软件的创始者们是完全没有预想到的。同时，他还与中国若干数学、计算机及农业研究所建立了紧密的合作关系。就像种子一经播下，便迅速成长一样，SCILAB 软件很快就在中国交到了许多朋友。2001 年、2002 年连续两届的 SCILAB 软件研讨会分别在 LIAMA 和上海复旦大学成功召开。“2002 年 SCILAB 竞赛”在众多中国大学的参与和支持下顺利举行，并借在北京举行的中欧信息论坛之际举行了正式的颁奖仪式。

SCILAB 软件在中国的发展生机勃勃，然而需要做的事情还很多，我非常赞同胡包钢先生的观点：SCILAB 软件在中国的经历仅仅是开始，其后将会更加精彩。现在，SCILAB 软件正处在一个充满机遇的十字路口：LIAMA 的几任主任积极努力促进中法的关系和友谊；教育、研究和工业之间的相互影响、渗透日益增强；数学建模在众多的科技领域中扮演着重要的角色。因此向中国读

者，不论是学生、学者还是工程师，介绍 SCILAB 这样一个软件着实是一个值得称赞的想法，胡包钢先生组织编写的这本书正好符合这一需要。

法国国立信息与自动化研究院(INRIA)
院长，首席执行官 Bernard Larrouturou

2002 年 8 月 29 日

前言

欢迎你，具有自由精神的科学计算软件
——SCILAB*

SCILAB是以法国国立信息与自动化研究院(INRIA)的科学家为主共同开发的“开放源码”式科学计算软件。SCILAB一词来源于英文“scientific laboratory”(科学实验室)词头的合并。

科学计算(如加、减、乘、除、微积分、逻辑推理等)是计算机应用的主要内容之一，并已经与实验研究、理论计算并列成为3大科学方法。[①] 以完成科学计算为目的的应用软件可以称为科学计算软件。在此，应该对“科学计算软件”有更为广义的理解。除了科学问题方面的计算，它同样适用于各种工程技术、金融、经济等方面的应用。目前这类软件多数是以数值计算形式为主，然而基于符号计算(如公式推导)的应用软件也变得日益普及。此外，科学计算可视化同样是该类软件的重要涵盖内容。

---

* SCILAB(c)INRIA-ENPC。

① 石钟慈，第三种科学方法——计算机时代的科学计算，北京：清华大学出版社，广州：暨南大学出版社，2000

所谓“开放源码(open-source)”是指该软件的全部源码程序是对外公开的。通常源码程序是商业软件中最为核心的商业机密,因为只有源码程序才能真正完整地表达或反映出软件开发中的精髓——软件设计思想。“开放源码”软件的一个重要特征是可以被用户免费自由地获取和应用。[①] SCILAB 正是这样一个可以在网站上免费下载、自由使用的科学计算软件。[②]

2001 年 4 月 9 日至 11 日,中国科学院自动化研究所中法信息、自动化与应用数学联合实验室(LIAMA)与法国国立信息与自动化研究院在北京联合举办了“中法科学计算自由软件—SCILAB 研讨会”。会议得到了中国科技部“863”高技术项目计划的大力支持并取得了圆满成功。但是目前的 SCILAB 与当前流行的商业科学计算软件 MATLAB 相比还有着一定的差距。就软件功能而论,一些中国用户可能不会选择 SCILAB。那么我们为什么要不遗余力地推广 SCILAB 呢?这固然有我们工作职责范围内的原因:中法实验室(LIAMA)应该为促进中法两国在信息、计算机领域中的交流与合作做出实际贡献。然而,在中国推广 SCILAB 软件,更主要还是由于以下几个方面的考虑。

1. 弘扬科学研究中的自由与共创精神

“科学无国境。”

“科学研究造福人类。”

“科学成果是人类共同努力的结晶。”

以上几点表明,科学本身充分体现了自由与共创精神。然而

① 当然用户将受到软件开发者预先规定的使用协议的限制。但是这些协议通常是针对不正当(如排他式)使用该软件而制定的。有关 SCILAB 软件使用协议请参见本书附录 B。

② SCILAB 主页:http://www-rocq.inria.fr/scilab/ 在中法实验室镜页:http://Lama.ia.ac.cn/scilab/

大多数技术发展总是与商业利益相关联。人类社会正是在兴趣与利益这样两个基本原动力作用下进步的。虽然目前的计算机应用是以商业化软件为主，但我们不应该忘记计算机软件同时具有科学方面的属性。

在这急功近利、技术至上、强者一统天下的时代，SCILAB 犹如一股春风吹拂大地。SCILAB 所表达的内涵已经超过软件本身，它在科学中体现的人文精神值得现代社会中的人们深思。我们不认为“开放源码”是软件业的惟一正确发展模式。自由精神（在此仅指非商业目的方面的自由精神）与商业利益是共辅的，这好比科学与技术之树，两者是一体生长的（见图 1）。科学对应植物的根系；技术代表树木枝干。当“商业价值”指标由下向上增长时，“自由精神”则是反向的。两者虽有竞争（如资源或养分），但更

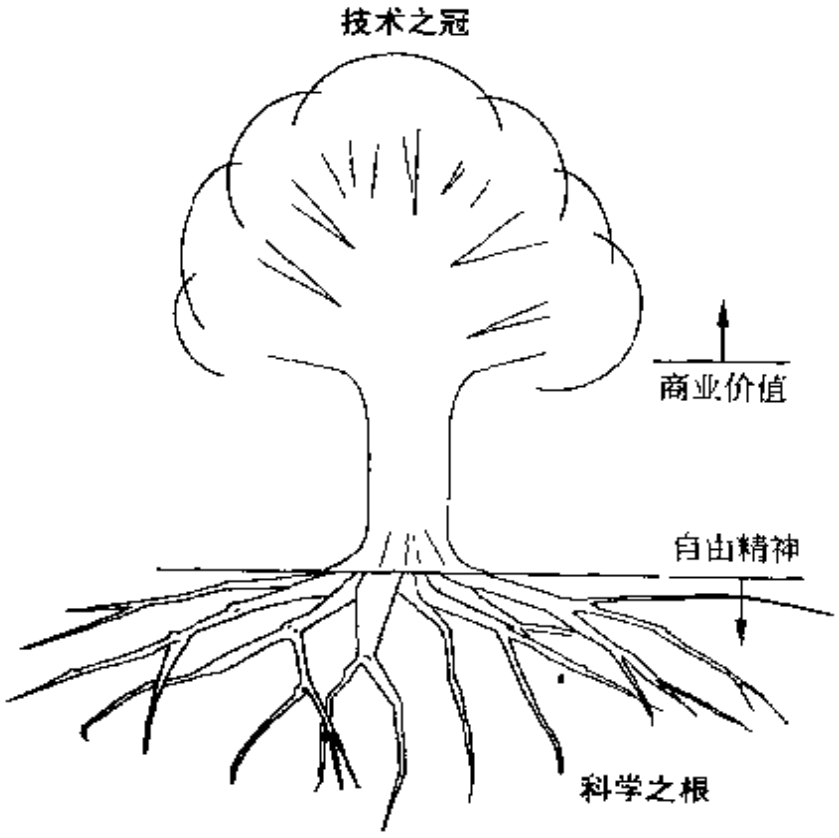


图 1 科学与技术发展的“树木说”

主要的是合作。只有根深叶茂，树木才能茁壮成长。自由需求开放，利益趋向封闭。纯商业软件一统天下的结果，难免会束缚计算机的普及应用和向更高水平的发展。可以理解，最为基本的、通用的科学计算软件部分应该走向自由与共享。只有这样才能最大限度地提高计算机应用的普及程度，使具有高新内容的商业软件业兴旺发达。另一方面，共享的结果可以汇集更广泛的智力，使软件不断趋于尽善尽美的水平。如果将软件业的发展比喻为飞鸟翱翔，则“开放源码”与“封闭源码”软件分别代表飞鸟的两翼（见图 2）。只有双翅奋扬，方可搏击长空。

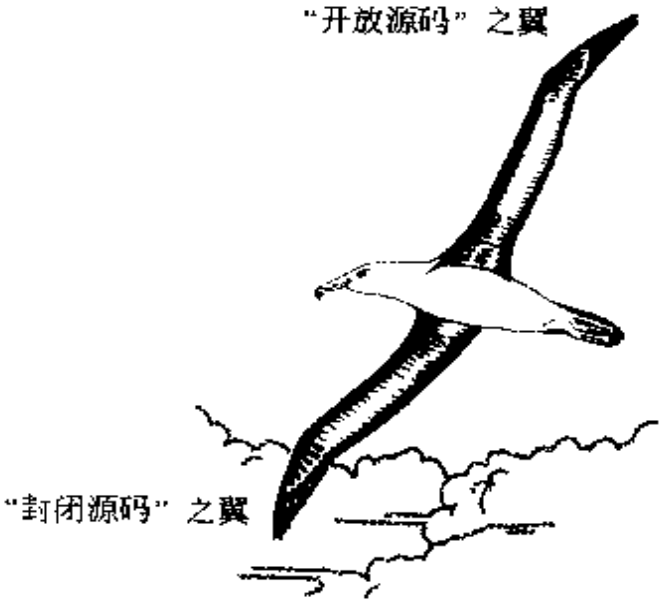


图 2　软件业发展的“鸟翼说”

2. 促进科学计算软件在我国教育中的普及应用

近几年来，我国教育界中应用科学计算软件的普及增长之势相当可观，这一点可以从计算机图书市场中相关图书的出版与发行情况估计出来。尽管如此，我国在这方面的发展仅仅是刚刚起步，普及科学计算软件在教育中的应用将为更新传统的教学方式

及其教学内容带来突破性的进展。例如，在物理学教学中，应用科学计算软件可以在计算机屏幕上形象地模拟皮球的自由落下、弹起、直至静止的动态全过程；对高中解析几何的学习可以借助有关软件以加深学生的理解。

如同正确对待使用计算器的问题一样，在一定阶段、以适当的方法引入科学计算软件并不会影响学生的抽象思维、手工推导过程的训练。不同于便携计算器操作（多数是一种机械方式输入过程），科学计算软件应用中更需要对知识的灵巧运用。借助科学计算软件开展的教学实验将为培养学生分析问题和解决问题的能力提供更为广阔的空间。在完成作业或解决实际问题时，同学们可以用计算机的各种图形方式给出结果。亲手应用科学计算软件可以大大提高学生们的学习兴趣和掌握科学计算工具的本领。我们认为，科学计算软件是可以在高中学习阶段中引入。而在理工科大学中，这些软件应成为学校教学计算机应用中的必备工具，这将是未来发展的必然趋势。

3. 缓解计算机用户资金短缺的问题

随着科学技术的普及发展和计算机应用进入各个领域，使用科学计算软件已经成为越来越常规的工作。在这样的背景下，MATLAB（主要基于数值式计算）、MAPLE（主要基于符号式计算）等科学计算软件得到了迅速发展和应用。正像计算机的许多发展已经超出我们的最初想象一样，科学计算软件的广阔发展前景同样是我们不可低估的。

中国应用科学计算软件的未来市场是巨大的。如果认为该类软件引入高中教学的想法是可取的，那么，仅这一项的市场就是相当可观的。然而，中国的教育机构面临着严重的资金短缺问题。由于国度间经济发展水平的差异，由西方工业国家开发出来的商用软件价格，国人很难承受。以 MATLAB 软件为例，目前主体部

分的价格要 1 万多元人民币(已超过一般 PC 机的价格),每一个工具箱需 3000 多元(工具箱是根据具体应用领域而选择的,MATLAB 可提供 20 多个工具箱)。以上价格是为学校提供的折扣价。若是非教育部门购买,其价格将更高。SCILAB 无疑为学校解决资金短缺问题提供了一条很好的出路。目前版本的 SCILAB 已经能够胜任科学计算中基本的数值计算并具可视化功能,完全可以满足高中以及大学教学内容的要求。为此,我们要感谢法国科学家们,以及所有为"开放源码"事业作出贡献的人们。

4. 倡导合法、规范性使用软件的风气与行为

合法、规范性地使用他人脑力劳动成果(如作品、专利、软件等)不仅是国家的政策,而且应该作为国人的一种理念予以倡导。只有这样,创新的源泉才能不被阻断。应该认识到,合法使用软件如果在社会中形不成主流,那么中国软件业将无法实现快速增长的目标。在教育界中推广、鼓励应用"开放源码"软件不单单是出于经济上的考虑,它将对学生建立合法使用软件意识、养成规范行为习惯有着潜移默化的作用。如同生态、环保教育一样,"知识产权"的教育与实践也应该从学生时代抓起。

在这些教育中,老师将是关键。当国内许多大学已经将软件应用作为教学与科研的基本工具时,我们学校的领导与老师有责任共同努力营造合法使用软件的环境与氛围。选用 SCILAB 的一个重要理由是:"我要自豪地将我在 SCILAB 上编写的作品标注上自主版权"。[①] 用户自行开发的 SCILAB 工具箱、应用软件、用户界面等,可作为独立的软件在 SCILAB 平台上运行,是完全属于用户自主产权的软件。用户(包括商业用户)可以按照任何方

① 请参考本书第 10 章中"关于 SCILAB 版权协议的说明"一节。

式处置这些软件。[①] 其中不仅能够完全保留“知识产权”，还可以根据“商业利益”进行销售。在这样的情况下，为什么不将我们开发的教学或科研成果建立在一个合法的平台上呢？

5. 培养人才、推动中国软件业的发展

人们常说，中国人适合于做软件。这有两个含义：一是中国人喜欢并且善于软件开发方面的工作，美国硅谷中有大量华人从事软件行业即证明了这一点；另一方面，计算机软件业属于劳动密集型产业，相对于其他传统工业而言，软件开发对场地、设备等诸多硬件条件的要求是相当少的，因此它特别适合于在中国这样的人口众多的国家中发展。作为“朝阳工业”的软件行业可以为人们提供大量的就业机会。“开放源码”方式将从技术的最基本层次为中国的软件业提供便利、可行的发展机遇。

“十年树木，百年树人”。这说明人才的培养难于实物的建立。教育领域应该成为软件业发展的最好起点。我们不怀疑中国能够培育出具有国际竞争能力的强大软件人员队伍与企业，使中国在未来多极化的国际软件市场中占有一席之地。我们更希望看到，中国的教育能够早日造就出世界级的计算机科学家。至少，在未来自由、共创的软件宝库中，如果能够看到众多来自中国的奉献，将会使国人真正感到自豪。

作为信息领域里的研究工作者，我愿意将微软等商家的口号“用正版、我自豪”改为“用正版、我坦然。创开源、我自豪”。

本书的出版正是基于以上的思考。在此我们衷心感谢法国研究人员的大力协助。这里特别要提到的名字是：M. Goursat, S.

---

① 在 SCILAB 平台上开发产品的“商业利益”一般只限于该产品范围，不包括 SCILAB 自身。这样使开发商与用户在降低产品成本方面都能获益。特别是在某些产品需求用户购买诸如 MATLAB 为软件平台的情况下。

Steer、C. Gomez, J.-P. Chancelier, F. Delebecque, R. Nikoukhah。他们的协助始于2001年的研讨会，我们很为他们这种超越种族、国界，不带任何功利性的合作与支持而感动。[①] 同时，我们还要感谢清华大学的孙增圻教授、江苏理工大学的黄建文教授在研讨会上做的专题讲座。可以相信，SCILAB所包含的"自由，共享，协作，同创"的科学精神已经得到并将继续会得到中国学者的热烈欢迎。在与中法信息、自动化与应用数学联合实验室(LIAMA)法方主任M. Jaeger以及J.F. Barczi研究员关于自由软件的讨论中我们受益匪浅。

值此机会，我们特别要感谢中法两国研究机构有关领导人员富有远见的支持。中国国家科技部副部长马颂德博士，与法国国立信息与自动化研究院院长兼执行总裁B. Larrouturou博士分别为本书撰写了序言。在促进SCILAB合作与交流中的人士中，还包括法国国立信息与自动化研究院副院长G. Kahn教授，国际合作部主任S. Grumbach（郭清溪）博士。中国方面有中国科学院自动化研究所所长谭铁牛博士与原国家"863"项目计算机主题首席科学家高文博士。再有，法国驻中国大使馆的有关人员也长期给予了支持，他们之中有法国驻中国大使毛磊(P. Morel)先生[②]、科技参赞卜来世(D. Blaise)先生[③]、米楠(A. Mynard)先生、夏良(G. Chalant)教授和张丽丽(L. Zhang)博士，在此一并对他们表示感谢。

本书第1,5,6,9章由赵星编写，第3,4章由康孟珍编写，第2,7, 8,10,11章以及全部附录部分由胡包钢编写。我们选择编

① 法国研究人员对我们的支持与帮助体现在本书的每一个章节。

② 每一位认识毛磊大使的中国人都会为他那富有哲理的谈吐和优美的人格魅力所打动。

③ 2001年中法SCILAB研讨会结束后，Blaise先生举行家宴特别邀请与会的部分中法人员，以祝贺会议成功。这样到家的全力支持令人难忘。

写，而非直接翻译 SCILAB 资料，是希望本书能够更适于中国读者阅读，包括高中以上的学生。书中难免有错误和不当之处，敬请读者指正。

胡包钢

2002 年春，于北京中关村

关于本书封面中红嘴海鸟标志的说明：

众所周知，Linux 软件有一个人见人爱的动物标志：黄嘴企鹅（penguin）。对于 SCILAB，本人建议使用英文称为 Puffin 的红嘴海鸟作为标志。这可爱动人的红嘴海鸟不仅能够展翅海天之间，还能直穿水下捕食小鱼（在加拿大纽芬兰省大西洋海岛上看到的这一奇观使我难以忘记）。我们欢迎用户自由使用该标志，用户可以不必注明该作品的出处。如若注明，敬请不要与以下内容相违：

Puffin for SCILAB

Designed by Who's Hu

第 1 章　概述 …… 1

1.1　SCILAB 简介 …… 1

1.2　怎样安装 SCILAB …… 2

1.3　SCILAB 的软件构成 …… 3

第 2 章　SCILAB 操作入门 …… 5

2.1　SCILAB 主窗口介绍 …… 5

2.2　如何学习 SCILAB …… 7

2.2.1　通过实例演示学习 SCILAB …… 7

2.2.2　应用“日志记录”学习 SCILAB …… 8

2.2.3　通过帮助方式学习 SCILAB …… 9

2.2.4　通过其他 MATLAB 教程资料学习 SCILAB …… 12

2.2.5　通过本书光盘中的文档资料学习 SCILAB …… 14

2.3　SCILAB 中的基本操作 …… 14

2.3.1　SCILAB 中的文件操作 …… 14

2.3.2　SCILAB 中预先规定的对象 …… 15

2.3.3 界面层次的控制操作 …… 17
2.3.4 SCILAB 主窗口中的快速键操作 …… 18
2.4 关于 SCILAB 的脚本文件 …… 19

第 3 章 数值运算 …… 23
3.1 矩阵的生成 …… 23
3.1.1 标量的生成 …… 23
3.1.2 向量的生成 …… 24
3.1.3 矩阵的生成 …… 26
3.1.4 特殊变量和常数 …… 28
3.2 基本数值运算函数 …… 29
3.2.1 加法和减法 …… 29
3.2.2 乘法运算 …… 30
3.2.3 矩阵求逆与除法运算 …… 31
3.2.4 乘方运算 …… 34
3.2.5 矩阵转置 …… 35
3.3 与矩阵有关的函数 …… 36
3.3.1 矩阵重新定维 …… 36
3.3.2 矩阵提取 …… 37
3.3.3 特征值和特征向量 …… 40
3.4 稀疏矩阵 …… 41
3.4.1 创建稀疏矩阵 …… 42
3.4.2 稀疏矩阵的查看 …… 47
3.5 数据统计 …… 50
3.6 数值积分 …… 56
3.6.1 一重定积分 …… 56

3.6.2 二重定积分 …… 57
3.6.3 三重定积分 …… 59
3.7 优化计算 …… 61
3.7.1 数据拟合 …… 61
3.7.2 非线性优化 …… 63
3.8 插值与样条 …… 67
3.8.1 插值函数 …… 67
3.8.2 样条函数 …… 68

第 4 章 SCILAB 数据类型及程序设计 …… 71
4.1 数据类型 …… 71
4.1.1 布尔矩阵 …… 71
4.1.2 字符串矩阵 …… 73
4.1.3 多项式类型 …… 79
4.1.4 表(list)类型 …… 87
4.1.5 函数 …… 92
4.2 程序控制语句 …… 94
4.2.1 循环语句 …… 94
4.2.2 判断语句 …… 97
4.3 SCILAB 脚本文件应用 …… 99
4.3.1 脚本文件的建立与调试 …… 99
4.3.2 函数参数与工作空间 …… 102

第 5 章 计算结果可视化 …… 106
5.1 SCILAB 图形窗口介绍 …… 106
5.2 二维图形的绘制 …… 108

5.2.1 plot 指令 …… 108
5.2.2 plot2di 指令 …… 111
5.3 三维图形的绘制 …… 115
5.3.1 三维曲面的绘制 …… 115
5.3.2 三维曲线的绘制 …… 120
5.4 绘图全局参数的设定 …… 123
5.5 色图的设定 …… 128

第 6 章 SCILAB 与 C 或 FORTRAN 程序的接口 …… 131
6.1 应用动态链接 …… 131
6.1.1 动态链接 …… 132
6.1.2 调用动态链接程序 …… 133
6.2 接口程序 …… 134
6.2.1 建立一个接口程序 …… 135
6.2.2 编译接口程序 …… 139
6.3 建立动态链接库 …… 139

第 7 章 SCILAB 应用篇 …… 144
7.1 信号处理 …… 144
7.2 系统分析 …… 148
7.3 线性方程组求解 …… 152
7.4 非线性方程组求解 …… 157
7.5 线性规划 …… 161

第 8 章 Scicos——图形化动态模型仿真器 …… 165
8.1 Scicos 基本内容介绍 …… 165

8.2 Scicos 子模块库介绍 …… 169
8.3 应用 Scicos 子模块库的一般注意事项 …… 180
8.4 Scicos 中的 3 类基本模块 …… 182
8.5 Scicos 应用实例 …… 184
8.6 Scicos 中的模块构造 …… 203

第 9 章 Tcl/Tk 在 SCILAB 中的应用 …… 209
9.1 Tcl 入门 …… 210
9.1.1 Tcl/Tk 解释器的安装及运行 …… 210
9.1.2 Tcl 基本指令 …… 211
9.1.3 Tcl 基本控制结构指令 …… 215
9.2 Tk 简介 …… 218
9.3 SCILAB 与 Tcl/Tk 的结合 …… 219
9.3.1 Tk_EvalFile 指令 …… 219
9.3.2 SCILABEval 指令 …… 220
9.3.3 Tk_GetVar 指令 …… 221
9.3.4 Tk_SetVar 指令 …… 221
9.4 一个应用实例:greenlab …… 222

第 10 章 关于自由软件及 SCILAB …… 226
10.1 为什么要讨论自由软件 …… 226
10.2 自由软件的发展简史 …… 227
10.3 自由软件的分类 …… 230
10.4 关于“Copyleft(开权)”的概念 …… 231
10.5 关于“开放源码”软件 …… 233
10.6 关于 SCILAB 版权协议的说明 …… 235

第 11 章 未完成的结束语 …… 239

附录 A SCILAB 光盘使用说明 …… 242
附录 B SCILAB 版权协议(中文译本) …… 248
附录 C SCILAB 部分函数指令表 …… 252

第1章

概　述

1.1 SCILAB 简介

SCILAB 是一个科学计算软件，它主要有两个功能：数值计算和计算结果可视化。SCILAB 数据类型丰富，可以很方便地实现各种矩阵运算。SCILAB 也能处理比数字矩阵复杂得多的对象，例如控制专业的多项式传递函数矩阵。在 SCILAB 中还定义了多项式、多项式矩阵和字符串矩阵，处理这些矩阵的方式与处理常向量和普通矩阵的方式相同。SCILAB 允许用户在线建立自定义函数。函数在 SCILAB 中被当作数据对象处理。例如在 SCILAB 中定义的函数可作为其他函数的输入或输出自变量。SCILAB 具有功能丰富的图形显示能力，可完成各种常规形式的计算结果可视化功能。

SCILAB 为用户提供如下计算和开放式编程环境：

(1) 多种容易操作的数据类型。

(2) 一个作为广泛计算基础的合理有效的基本函数集。

(3) 一个开放式编程环境，新的函数能很容易地被添加。Intersci 是一个有用的发布工具，通过它能建立接口，添加新的函

数及工具箱，例如增加新的 FORTRAN 代码和 C 代码到 SCILAB 中。

SCILAB 还包括一些应用于不同科学计算领域的工具箱，例如应用于数学建模、信号处理、网络分析、决策优化、线性与非线性控制等多个方面的工具箱。Scicos 工具箱允许图形定义和模拟复杂的连续和离散的混杂系统。

SCILAB 是一种解释性语言，能运行于 Windows，Linux 以及 UNIX 等操作系统环境下。鉴于目前我国大多数计算机用户使用 Windows 系统，因此本书主要介绍 SCILAB 的 Windows 版本的使用方法。

SCILAB 与目前流行的 MATLAB 软件起源相同，都源自于 Cleve Moler 于 1980 年开发的程序，因此它的功能与 MATLAB 软件相似，并且表达式的语法、函数的调用和大多数控制指令都类似。但由于这两个软件以后各自的独立发展，因此也有一些不同。

SCILAB 所带的文献主要包括用户指南（SCILAB 介绍）和 SCILAB 在线手册，还有一些针对特定工具箱的报告，如 Scicos（图形化动态模型仿真器）、Signal（信号处理工具箱）、Control（控制工具箱）和 Metanet（图形和网络工具）。

1.2 怎样安装 SCILAB

SCILAB 是一个开放源代码的自由软件，它的全部源代码程序和执行码程序可以从本书所附的光盘中得到（参见附录 A 中说明），也可以从 SCILAB 网站 http://www-rocq.inria.fr/scilab 上下载。SCILAB 的安装程序、说明文档、应用例子以及一些最新更新消息等都能从该网站上得到。

安装要求如下：

对于二进制版本：解压缩后运行 SCILAB 至少需要 40MB 的

空间(不包括源代码)。这个版本包括部分静态链接。

对于源程序版本:展开和安装(包括所有的源代码)SCILAB需要大约130MB的磁盘空间。用户需要X Windows(X11R4,X11R5或X11R6)、C编译器和FORTRAN编译器(例如f2c或g77)以及用于Windows系统的Visual C++。

安装SCILAB时,可以运行其安装文件SCILAB26.exe自动安装;也可以将磁盘文件解压缩后,直接复制到你的目录之下。卸载SCILAB时,对于自动安装的文件,通过卸载命令卸载;对于以直接复制方式安装的SCILAB,直接删除所复制的文件即可。

关于更多的安装信息,请看SCILAB的README文件。

1.3 SCILAB的软件构成

SCILAB由3个独立的部分组成:一个解释器、函数库(SCILAB程序)以及一个FORTRAN和C程序库。SCILAB的内容分若干个子目录集。在主目录SCIDIR中包含下列几个重要文件:启动文件scilab.star、版权文件notice.tex和configure文件。scilab.star是一个文本文件,其中配置了SCILAB的许多重要启动参数。例如由于SCILAB是用一个大堆栈来计算,因此可通过修改scilab.star多项式、多项式矩阵和传递函数矩阵文件中的一个重要参数newstacksize变量的值来调整堆栈尺寸。

在主目录下包含如下子目录:

(1) bin是可执行文件目录。在UNIX/Linux系统中执行文件名是SCILAB;在Windows95/NT中是runscilab.exe。SCILAB的可执行代码在UNIX/Linux上是scilex;在Windows95/NT上是scilex.exe。这个目录还包括用于打印SCILAB生成的Postscript和Latex文件的外壳程序。

(2) demos是演示程序目录。这个目录包含对应各种演示程

序的代码，这些代码对初学者很有用。文件 alldems. dem 可以通过“Demos”菜单项使用。大多数绘图命令通过简单的演示程序例子来加以说明。注意运行没有输入参数的图形函数时，将给出使用这个函数的例子，例如 plot2d()显示应用 plot2d 函数的例子。

(3) examples 目录包含许多 SCILAB 例子。其中包括怎样利用外部程序建立 SCILAB 的动态链接库的例子。链接外部程序到 SCILAB 使用动态链接或 intersci，本书第 6 章有专门叙述。

(4) doc 是 SCILAB 文档目录，包括 Latex，dvi 和 Postscript 文件。这个文档是 SCIDIR/DOC/intro/intro. tex.

(5) geci 包含 GeCI 的源代码和二进制文件。GeCI 是一个交互式管理器，用于管理远程软件的执行和程序之间的信息交换。它使在网络上利用几台计算机组成一台虚拟计算机成为可能。GeCI 用于连接 Xmetaner 和 SCILAB。

(6) pvm3 包含 PVM3(Parallel Vitual Machine)的源代码和二进制文件，PVM 版本 3 是另一种交互式通信管理器。

(7) libs 包含 SCILAB 库(编译过的代码)。

SCILAB 操作入门

2.1 SCILAB 主窗口介绍

安装完后在计算机界面上启动 SCILAB，将弹出 SCILAB 主窗口（见图 2.1）。该窗口是 SCILAB 用户进行人机交互的主要界面，也是命令与数据的输入输出窗口。未作特殊说明，以下介绍的操作都是在 Windows 平台上的 SCILAB 2.6 版本操作，在 Linux 或 UNIX 上的操作与显示情况会略有不同。

在窗口上方包括 6 个下拉菜单。它们的含义分别如下：

【File】	【文件】
Getf	读函数
Exec	执行文件
Save	存文件
Load	读文件
Change Directory	更改当前目录
GeT current Directory	读当前目录
Exit	退出 SCILAB
【Edit】	【编辑】

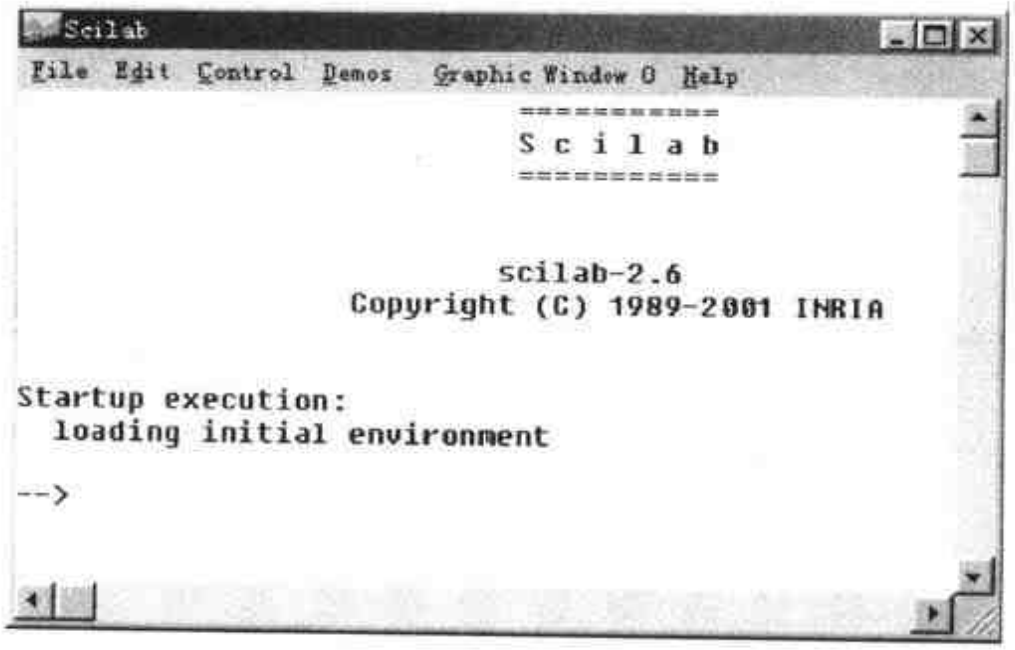


图 2.1 SCILAB 主窗口

Copy to Clipboad	复制
Paste	粘贴
Choose Font	选择字体大小
History	记录
【Control】	【控制】
Restart	启动 SCILAB 初始状态（同时清除全部用户变量）
Pause	暂停(指令界面上升一层)
REsume	回复(指令界面下降一层)
Abort	取消运行(指令界面下降到 SCILAB 初始层)
Interupt	中止运行(同时指令界面上升一层)
【Demos】	【演示】
【Graphic Windows 0】	【图形窗口-序列号 0】

Set(Create) Window	设置窗口
Raise(Create) Window	弹出图形窗口
Delete Graphic Window	取消图形窗口
+ (increase current num)	增加目前窗口序列号
− (decrease current num)	降低目前窗口序列号
Clear Current Window	清除目前窗口内容
【Help】	【帮助】
Help Dialog	列表对话帮助
Topic	指令查询帮助
Apropos	关键字查询帮助

图 2.1 中最下一行的提示符“--)”是 SCILAB 命令的输入部位。对于下拉菜单键中的若干项目，用户同样可通过键入命令完成操作。例如键入“exit”命令后，系统将自动退出 SCILAB。

2.2 如何学习 SCILAB

2.2.1 通过实例演示学习 SCILAB

熟悉 SCILAB 的最基本操作可以从对下拉菜单【Demos】的实例演示开始。这是一个多层选择窗口的操作演示方式。图 2.2 是根据 SCILAB 子函数库分类的选择对话窗口。选择“Introduction to SCILAB”，单击“OK”键，此时 SCILAB 主窗口自动显示执行该示例命令与计算结果。这些命令与计算结果对于用户理解和学习 SCILAB 有很大帮助。**当主窗口出现提示符“>>”时，表明 SCILAB 示例演示程序处于暂停状态(应使用mode(7)实现而不是pause)，目的是让用户观测示例执行到目前的结果。用户必须在主窗口下按“回车”(Enter 或 Return)键，示例才能继续程序的运行，同时，示例可能还会弹出图形窗口。当最终主窗口出现提示符**

"-->"时,或退回到演示选择对话窗口,表明该演示程序结束。

在SCILAB主目录下有一个称为"demos"的子目录,该子目录又包含若干个子目录,其中,包括以"*.dem"为扩展名的文件。用户可以用纯文本编辑器打开这些文件(也可以称为脚本文件)进行实例学习。

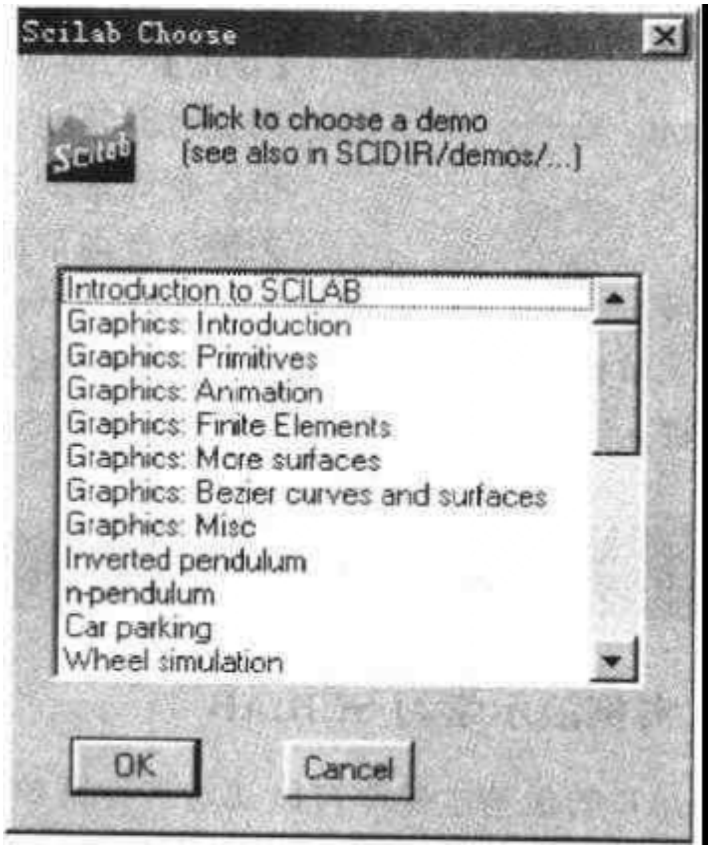


图2.2 SCILAB实例演示选择对话窗口

2.2.2 应用"日志记录"学习SCILAB

当采用上述SCILAB运行示例演示时,在SCILAB主窗口上经常会显示出大量指令与数据内容。有时其内容已经超出SCILAB主窗口可以追溯的部分,致使用户的跟踪学习出现困难。此时,可以应用"日志记录"方式检查或学习全部的示例指令与曾

经显示的数据。示例操作如下：

```
-->diary('myrecord')
-->a=[12.34    45.74    139.3 0.65]
 a    =
!     12.34   45.74   139.3        .65 !
-->sum(a)
 ans    =
  198.03
-->diary(0)
```

在上面的示例中主要先用 diary 指令在当前子目录下生成一个纯文本文件“myrecord”，该文件记录所有在 SCILAB 主窗口中显示的内容，直到遇到“日志记录”中止指令“diary(0)”。此时，用户可以打开该文件阅读该示例中的全部文档方面的记录，这个记录对于用户回溯以往的计算数据也是十分有用的。此示例如同记录进行核算的账目清单。应用 SCILAB 不仅比应用普通计算器更为方便，也易于再次查询。

2.2.3 通过帮助方式学习 SCILAB

另一个重要的学习方式是应用【Help】帮助方式。例如在上面的实例演示中，对许多指令的具体内容不了解，比如 feval。这时可以另启动一个 SCILAB 主窗口（两个 SCILAB 交叉使用将更为方便），然后在指令输入部位键入

```
--> help feval
```

此时，SCILAB 弹出一个窗口，一般包含该指令的应用解释与示例。对于这个示例，可以用鼠标器中的左键选择出示例的指令部分，右击选择“复制”，回到 SCILAB 主窗口，右击选择“粘贴”，按“Enter”键后完成示例执行任务。例如“feval”帮助文档中给

出的实例：

```
--> deff('[z]=f(x,y)','z=x^2+y^2');
--> feval(1:10,1:5,f)
 ans  =

!    2.    5.    10.   17.   26.  !
!    5.    8.    13.   20.   29.  !
!    10.   13.   18.   25.   34.  !
!    17.   20.   25.   32.   41.  !
!    26.   29.   34.   41.   50.  !
!    37.   40.   45.   52.   61.  !
!    50.   53.   58.   65.   74.  !
!    65.   68.   73.   80.   89.  !
!    82.   85.   90.   97.   106. !
!    101.  104.  109.  116.  125. !
```

应用该方式来学习将大大加快用户对指令的理解。“选择”、“复制”和“粘贴”这样的操作过程经常可以在SCILAB应用中借鉴使用。在帮助文档结尾处，通常有关于与当前指令相关的其他指令，这将有助于用户快速查询相关指令。

另一种方式是打开【Help】菜单，选择“Help Dialog”后，弹出“帮助选择对话窗口”(见图2.3)。其中可以根据SCILAB子功能库的范围选择特定的指令。或者在“apropos”中键入指令的关键字，例如“format”。SCILAB将在帮助文档中找到包含“format”一词的全部相关指令。

下面对“Help”中的指令调用格式(CALLING SEQUENCE)进行说明，以“plot2d”为例。该指令在帮助中给出以下3种格式：

```
plot2d([x],y)
plot2d([x],y,<opt_args>)
plot2d([logflag],x,y,[style,strf,leg,rect,nax])
```

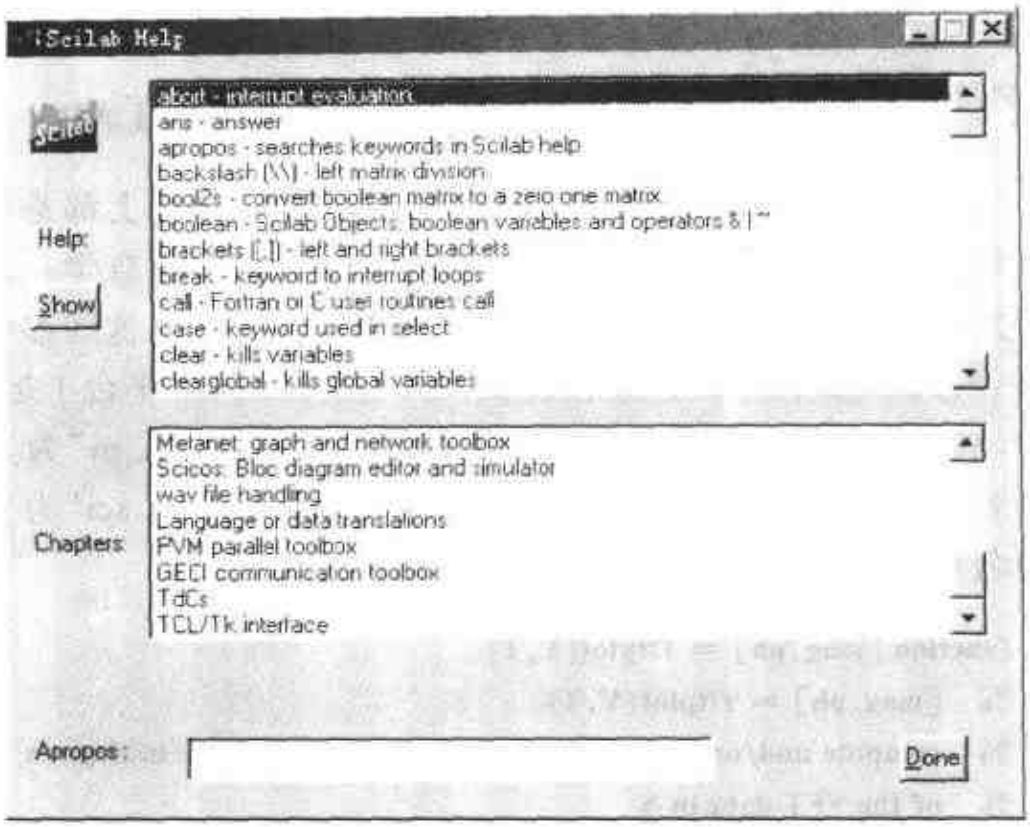


图 2.3 SCILAB 帮助选择对话窗口

首先，可以看出，这是个有输入变量的指令（事实上，如果全部输入变量未给定，SCILAB 将显示给定的特例）。其中，变量 y 是必须输入的数据；方括号“[ ]”代表其中的变量用户可以选择输入，当用户没有给定输入时，SCILAB 按照缺省值给定；尖括号“<>”也表明可以缺省输入，只是尖括号内的变量名称及个数未给出（**opt_args** 只是该部分变量的代名词），需要用户进一步阅读帮助文档。

其次，帮助文档对每一个变量的含义会给予解释。当规定变量是字符串时，必须用双引号表示，例如 **plot2d("nl" ,x,y)** 表明 **logflag** 是字符串变量，该输入给定横坐标是常规方式，而纵坐标是对数方式。该变量的缺省值是“**nn**”，即横纵坐标都为常规方式。

最后，当缺省变量未给定时，其跟随的逗号要相应取消。例如

应该使用 plot2d(x,y),而不是 plot2d(,x,y)。

2.2.4 通过其他 MATLAB 教程资料学习 SCILAB

SCILAB 与 MATLAB 的许多指令在名称与功能上都是相同的。目前,已经有大量的 MATLAB 参考书可以用于自学。本书参考文献中推荐了几部好的 MATLAB 中文参考书。这些参考书上的 MATLAB 程序经过适当转换,可以在 SCILAB 平台上运行。其中,SCILAB 提供了将 MATLAB 脚本文件(以"*.m"为文件扩展名)自动转换为SCILAB的脚本文件格式(以"*.sci"为文件扩展名)指令。例如在当前目录下有"fftplot.m"文件:

```
function [mag,ph] = fftplot(Y,T)
%   [mag,ph] = fftplot(Y,T)
%   compute and/or plot the magnitude in dB and phase in degrees
%   of the FFT data in Y
%   T = sampling period, if this parameter is not given it is assumed
%       to be equal to one.
%   if called with both output arguments, i.e. / [m,p] = fftplot(Y)
%   the magnitude and phase are returned in m and prespectively
%   and no plot is generated
%   if called with one output argument, i.e.   / m = fftplot(Y)
%   the magnitude of Y in dB is returned and a plot of the magnitude
%   of Y is generated in the range 0 s w s pi
lab = ['DIGITAL FREQUENCY / pi'
       '    FREQUENCY HERTZ  '];
ly = log(Y);
mag = 20 * real(ly) / log(10);
if nargout == 2
                    ph = imag(ly);
elseif nargout <= 1
                    n = max(size(Y))/2;
```

```
        f = (0:n-1)/n;
        if nargin == 2, f = 0.5 * f/T; end
        plot(f,mag(1:n)),title('MAGNITUDE'),ylabel('dB'),
        xlabel(lab(nargin,:))
end
```

应用 mfile2sci 转换指令，SCILAB 将在同一目录下自动生成一个称为“fftplot.sci”的文本文件。该文件可以在 SCILAB 平台上运行。如下的一段 SCILAB 脚本程序先将源文件转换为 SCILAB 函数，然后调用该 SCILAB 函数，并得到图 2.4。

```
mfile2sci fftplot.m
    getf fftplot.sci
    t=0:0.1:100;
    y=fft(sin(t)+sin(5*t)+3*sin(3*t),-1);
    fftplot(y,1);
```

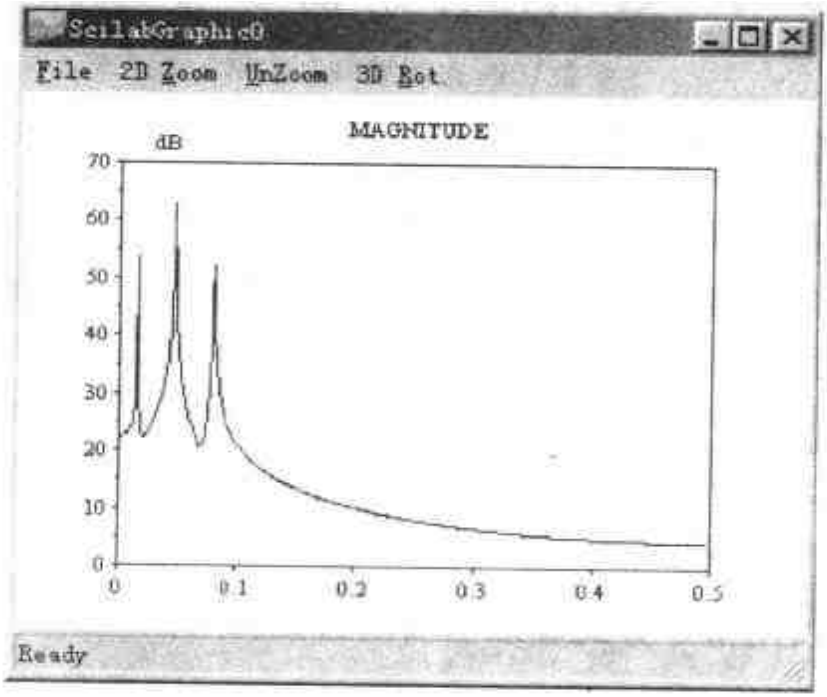


图 2.4 运行 fftplot(y,1)函数显示的图形

从上面的示例中可以看到，SCILAB可以较好地自动完成文件的转换工作，但是在图形等方面的指令转换上，有时仍然需要用户的手工操作。更好的学习方式还是用户读懂MATLAB程序后，根据SCILAB指令方式逐句手工编程。SCILAB还提供了将其脚本文件自动转换为MATLAB文件(应用指令 sci2mfile)，以及将以MATLAB编写的工具箱中的众多文件同时转换为SCILAB文件(应用指令 translatepaths)的功能。

2.2.5 通过本书光盘中的文档资料学习SCILAB

本书提供光盘一张。其中包括由INRIA与ENPC提供的大量学习SCILAB的英文文档资料。有关文档的基本内容请见本书附录A。

2.3 SCILAB中的基本操作

2.3.1 SCILAB中的文件操作

SCILAB主窗口中的下拉菜单【File】用于完成SCILAB的文件操作。首先，选择“GeT current Directory”(读当前目录)来显示当前的目录：

```
-->getcwd()
 ans    =
 C:\WINDOWS\Desktop
```

当然，也可以用指令方式显示当前的目录：

```
-->pwd
 ans    =
 C:\WINDOWS\Desktop
```

但是 getcwd()与 pwd 作为操作指令的不同之处是前者可以赋值(如 a= getcwd()),后者则不能。

一般用户需要在自定义的子目录下调用自编程序,这就需要更改当前目录。方法可以选择下拉菜单中的操作命令或输入命令:

```
-)chdir('d:\usrdir')
```

执行文件也可以采用类似的操作命令,比如启动 SCILAB 中的示例演示程序:

```
-)exec('SCI/demos/alldems.dem');
```

这里 SCI 已经是 SCILAB 预先规定的变量(见下节),指向 SCILAB 软件的主目录。

2.3.2 SCILAB 中预先规定的对象

在 SCILAB 中已经预先规定了一些对象。这些对象主要是一些变量、常数等。用户在自定义变量名称时,不可以与已规定的对象名称重复。当用户启动 SCILAB 之后,或者选择"Restart"操作之后,可以看到这些对象名称:

```
-)who
your variables are...
 startup    ierr       demolist   %scicos_display_mode
 scicos_pal          %scicos_menu          %scicos_
short        %helps
 MSDOS      home       PWD        TMPDIR
  percentlib          soundlib
 xdesslib  utilslib   tdcslib    siglib   s2flib   roblib   optlib
 metalib   elemlib    commlib    polylib  autolib  armalib  alglib
 intlib    mtlblib    WSCI       SCI      %F       %T       %z
```

```
%s      %nan     %inf     $       %t      %f      %eps
%io     %i       %e
using     6105 elements out of   1000000.
      and          46 variables out of       1791
```

通过键入某个对象名称可以观察到这些对象的内容，例如：

```
-->startup, typeof(startup)
 startup    =
     []
 ans    =
 constant
```

可以看到 startup 是一个空集，由 typeof 指令得知这是个常数类型对象。

另一种方式是可以应用 whos 指令来观测全部对象的名称和内容：

```
-->whos();
Name                    Type        Size      Bytes
whos                  function                6232
ans                     string      1 by 1      56
typeof                function                4104
startup               constant      0 by 0      24
ierr                  constant      1 by 1      24
demolist                string      28 by 2     4704
%scicos_display_mode  constant      1 by 1      24
scicos_pal              string      7 by 2      1192
%scicos_menu             list                   2304
        ---------------------------------------
```

应用 clear 指令可以消除用户定义的全部对象，而无法消除 SCILAB 预先规定的对象。

2.3.3 界面层次的控制操作

SCILAB主窗口的下拉菜单【Control】中包括了一些界面层次的控制操作。SCILAB中常见的提示符可以称为初始层界面。根据程序的需要,SCILAB提供了如下多层界面的操作:

```
-)a=rand(1:4)
 a    =
!    .7560439      .0002211      .3303271      .6653811 !
-)pause
-1->b=1-a
 b    =
!    .2439561      .9997789      .6696729      .3346189 !
-1->resume
-)
```

在主窗口键入 **pause** 命令(也可以在下拉菜单中进行选择)后,SCILAB进入高一层界面,提示符变为"-1->"。在该层可以完成一些程序中间结果的计算验证工作。完成工作后键入Resume(或Return)命令退回初始层界面。**应该注意,底层界面的变量可以被高层使用;而高层界面的变量无法传递到底层。**例如在上例中,初始层中不能直接应用向量 b 的结果。这样就确保了主窗口界面下的程序能够正常运行,而不受程序运行中在高层界面进行人工检验时引起的数据干扰的影响。用户还可以应用"Ctrl+c"双键操作嵌套进入高层界面。而一个 **abort** 命令则可一次退回 SCILAB 的初始层界面。

如果选择下拉菜单中的"Restart"操作将使 SCILAB 重置初始状态,这不仅能确保退回 SCILAB 初始层界面,也将清除掉用户定义的所有变量或常数等对象。**注意,当 SCILAB 在运行过程中出现错误而无法正常运行时,一般可以通过执行"Restart"命令**

来使SCILAB恢复正常运行。但是用户应该将自定义的全部变量及计算结果预先进行存储：

```
-)save('d:\usrdir\mydat.dat');
```

当执行完"Restart"命令之后，可以用如下命令将全部变量及计算结果调回SCILAB工作空间，以备用户使用。

```
-)load('d:\usrdir\mydat.dat');
```

2.3.4 SCILAB主窗口中的快速键操作

在SCILAB主窗口下，用户可以使用一些快速键进行命令操作。这些操作对于追溯曾经操作过的命令，进行编辑、运行等是十分方便的。表2.1给出了快速键的操作功能。

表2.1 SCILAB快速键的操作功能

键名	操作功能	键名	操作功能
↑	向前搜寻已经输入的命令	Delete	删除当前光标字符
↓	向后搜寻已经输入的命令	Esc	清除命令行的全部内容
←	光标向前移动一个字符	Ctrl + c	命令界面上升一层
→	光标向后移动一个字符	Ctrl + p	向前搜寻已经输入的命令
Home	使光标移到命令行首端	Ctrl + n	向后搜寻已经输入的命令
End	使光标移到命令行尾端	Ctrl + b	光标向前移动一个字符
Backspace	删除光标左边字符	Ctrl + f	光标向后移动一个字符

续表

键 名	操 作 功 能	键 名	操 作 功 能
Ctrl + a	使光标移到命令行首端	Ctrl + u	清除命令行的全部内容
Ctrl + e	使光标移到命令行尾端	Ctrl + k	删除当前光标及其右边的字符
Ctrl + h	删除光标左边字符	Ctrl + w	删除命令行的最后一个词
Ctrl + d	删除当前光标字符		

2.4 关于 SCILAB 的脚本文件

SCILAB 主窗口是一个人机交互方式的计算平台。对于比较简单和一次性计算问题，用户可通过命令方式去求解，以达到简单、快捷的效果。但是当求解问题规模较大时，应用所谓的“脚本文件”方式将是十分必要的。下面是一个典型的“脚本文件”：

```
//  File name: MyExample.sci
//  This file is to generate and plot a curve.
clear                                  // To clear the variables of SCILAB
xbasc()                                // To clear the graphic window
x=[0:999] * 4/1000; y= ...             // To make a data set for x, and
                                       // followed by
  sin(2 * x.^2);                       // two-line sentences to generate
                                       // the function
plot2d(x,y)                            // To plot the function
plot(x,y,"x","y","Plot of a function")  // To put the context
                                           // and caption
xgrid()                                // To add the grid to the grphics
// After running this file, you will see a graphic window showing a plot
```

```
// of the function, as well
//   as get the data of x and y.
```

将该文件保存在SCILAB主目录下，在SCILAB主窗口下，可以运行该文件：

```
->exec('d:\usrdir\MyExample.sci');
```

当运行完该文件后，SCILAB自动弹出一个图形窗口（见图2.5），其中包含函数方程 $y=\sin(2x^2)$ 曲线。下面是一些关于脚本文件的说明：

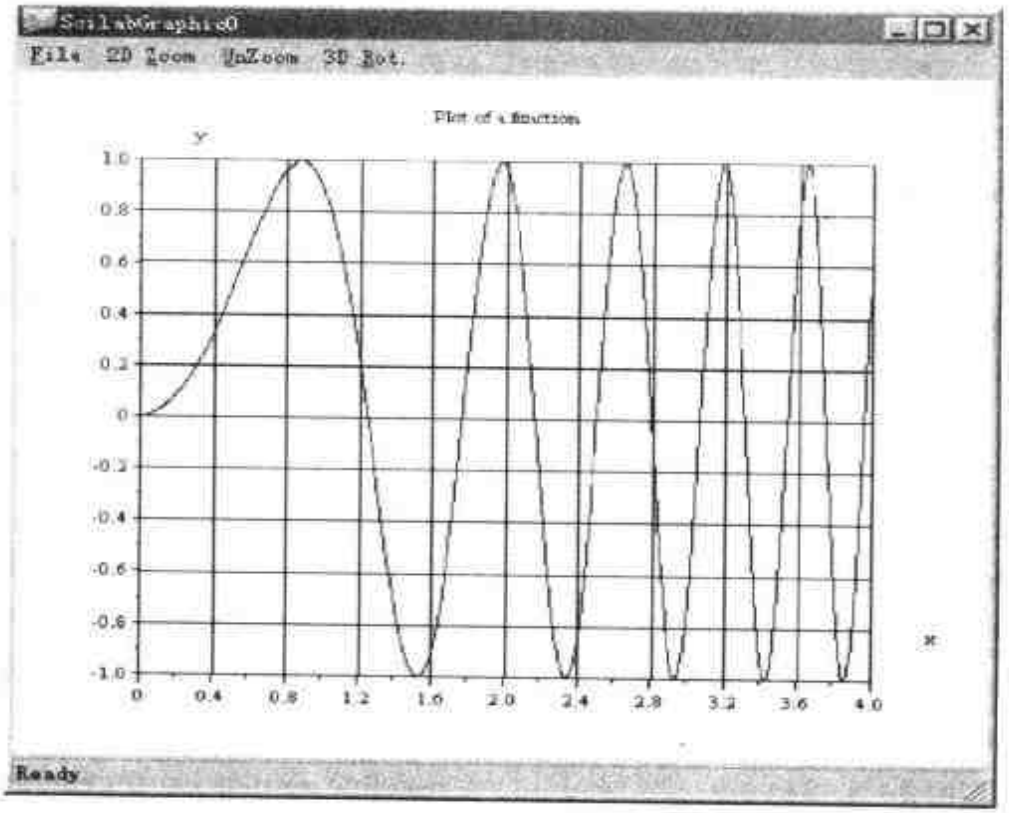


图 2.5 函数方程 $y=\sin(2x^2)$ 曲线图

(1) SCILAB脚本文件的扩展名一般是“*.sci”。

(2) SCILAB脚本文件是一个纯文本文件，可以使用任何一个文本编辑器进行编辑。由于目前SCILAB本身不带编辑器，

因此用户必须自己选择一种编辑器，例如 Windows 附件中的写字板(NOTEPAD. EXE)。附录 A 给出一个更好的软件编辑器"PFE"的下载网址。由于该软件具有快捷、方便的编辑功能，适合编辑 SCILAB 中纯文本程序文件，建议用户最好使用该编辑软件。该软件也是一个自由软件，用户可以非商业方式地免费分发和使用。

(3) SCILAB 脚本文件与直接使用主窗口中的命令是完全等效的。

(4) SCILAB 脚本文件中的注释内容是以"//"开始的。例如本例中的开始与结尾两行，以及各个命令后的注释。注释的目的是使用户便于理解程序的执行内容，取消上面全部注释，SCILAB 将得到同样的计算结果。

(5) SCILAB 允许空白行，该行不产生任何命令操作。

(6) SCILAB 允许空白符，空白符不影响命令操作。但是命令文字中要求连续，不能有空白符。

(7) SCILAB 命令一般是以分号";"作为结尾。当某一赋值命令没有该分号作结尾时，SCILAB 主窗口将显示全部计算结果。

(8) SCILAB 允许一行内包含多个命令。但是命令之间必须用分号";"分隔，或者用逗号","分隔。使用前者，SCILAB 主窗口不显示计算结果；使用后者，SCILAB 主窗口显示计算结果。

(9) 在 SCILAB 脚本文件编写中会发现有些命令不能在一行内完成，必须续行后完成。此时，可以用续行符"..."接在续行指令中的末尾，如上例中计算 y 值的情况。SCILAB 对于续行次数没有限制。

(10) SCILAB 脚本文件可以调用其他用户定义的脚本文件。其他脚本文件可以用函数形式写成。但是脚本文件不能调用自身

(不具备递归功能)。

(11) 在运行新文件时,一般建议对SCILAB工作空间中的变量及图形窗口进行清除处理,例如本例中,分别使用了**clear**与**xbasc()**命令,以防止原有变量或图形内容对新运行文件结果的干扰。

第3章

数值运算

SCILAB应用中的一个基本功能是数值运算。SCILAB以矩阵作为基本的运算单元，本章重点介绍与矩阵操作有关的内容。

3.1 矩阵的生成

SCILAB矩阵的通用形式是多维数值型数组。标量与向量都被看作特殊的矩阵：标量为只有一个元素(行数和列数都为1)的矩阵，向量为只有一行或一列(行数或列数为1)的矩阵。

产生一个矩阵最直接的方法是通过键盘输入赋值。通过键盘输入数据生成矩阵只需遵循以下规则：

(1) 在矩阵的每一行中元素之间以空格或者逗号隔开；

(2) 在矩阵的行与行之间以分号或回车键隔开；

(3) 整个矩阵包含在一对方括号中。

3.1.1 标量的生成

标量的生成很简单，只需要直接给一个变量赋值：

例 3.1 产生标量

```
-->a=5.3
                a    =
              5.3
```

由于标量为只有一个元素的矩阵，因此也可以把数据放在一对方括号中：

```
-->a=[2.5]
          a    =
          2.5
```

3.1.2 向量的生成

向量的生成有若干种方法，一类是直接输入数据，另一类是自动生成。直接输入数据时，行向量的各分量间用空格或者逗号隔开，列向量的分量间用分号隔开。

例 3.2 直接赋值产生向量

```
-->v=[1 2 3]  //生成一个行向量
       v    =
    !    1.    2.    3.  !
-->v=[1;2;3]  //生成一个列向量
       v    =
    !    1.  !
    !    2.  !
    !    3.  !
```

自动生成方式是利用“:”号生成均匀等份的向量。其格式为 v=j:k，j 和 k 不一定为整数，但是 j 必须小于 k，否则生成空向量。默认的元素间隔为 1。也可以设置向量步长，其格式为 v=j:i:k，其中 i 为步长。

例 3.3 利用":"生成向量

```
-->v=1:5
 v    =
!    1.    2.    3.    4.    5.  !
-->v=1:0.5:5
 v    =
!    1.    1. 5    2.    2. 5    3.    3. 5    4.    4.
5    5.  !
```

这种给向量赋值的方式在编程中比较常见。

第三种方式是利用 linspace 函数生成元素个数可以明确给定的等份向量,其调用格式如下:

```
[y]=linspace(d1,d2,n)
```

函数返回在 d1 与 d2 之间均匀分布的有 n 个元素值的行向量,n 的缺省值是 100。

例 3.4 利用 linspace 函数生成向量

```
-->v=linspace(1,100,10)
 v    =
!    1.    12.    23.    34.    45.    56.    67.    78.
89.    100.  !
```

当 n 缺省时,[y]=linspace(d1,d2) 在 d1 与 d2 之间自动生成有 100 个元素的线性等份向量。

如果需要对数等分向量,即元素的对数值线性等分的向量,SCILAB 提供 logspace 函数实现这一功能。函数调用格式为

```
[y]=logspace(d1,d2,n)
```

函数返回在 10^{d1} 与 10^{d2} 之间的 n 个元素构成的向量,这些元素以对数基为线性等分。n 的缺省值是 50。

例 3.5 利用 logspace 函数生成向量

```
-->v=logspace(1,3,3)
 v    =
!    10.     100.     1000.  !
```

3.1.3 矩阵的生成

简单矩阵生成的常用方式是直接从键盘输入矩阵。

例 3.6 生成一个 3×4 的矩阵：

在提示符下输入

```
-->m=[1,2,3;4,5,2;4,3,2]
```

屏幕输出为

```
 m    =
!    1.     2.     3.     4.  !
!    4.     5.     2.     8.  !
!    4.     3.     2.     9.  !
```

或者分行输入

```
-->m=[1 2 3 4
-->4 5 2 8
-->4 3 2 9]
```

输出结果一样：

```
m    =
!    1.     2.     3.     4.  !
!    4.     5.     2.     8.  !
!    4.     3.     2.     9.  !
```

在显示时 SCILAB 用“!”号对每一行定界。在以上两个输入形式中：第一种形式的行内元素用逗号隔开，行之间用分号隔开；

第二种形式的行内元素用空格隔开，行之间用回车符隔开，但所赋的数值内容是一样的。已生成的数据保存在 SCILAB 工作空间中，可以通过调用变量名"m"进行使用，例如显示、参与运算等。

常用矩阵的生成：

(1) 全 0 阵，即元素全为 0 的矩阵。函数名称为 zeros，调用格式如下：

zeros(m1,m2)：生成 m1×m2 的全 0 矩阵。

zeros(m1,m2,…,mn)：生成 m1×m2×…×m*n* 的全 0 矩阵。

zeros(A)：生成大小与矩阵 A 相同的全 0 阵。

zeros()：返回单个 0 值，相当于 **zeros(1,1)**。

(2) 全 1 阵，即元素全为 1 的矩阵。函数名称为 ones，调用格式如下：

ones(m1,m2)：生成 m1×m2 的全 1 矩阵。

ones(m1,m2,…,mn)：生成 m1×m2×…×m*n* 的全 1 矩阵。

ones(A)：生成大小与矩阵 A 相同的全 1 阵。

ones()：返回单个 1 值，相当于 ones(1,1)。

(3) 单位阵，即对角线元素为 1，其他元素为 0 的矩阵。函数名称为 eye，调用格式如下：

eye(m,n)：生成 m×n 的对角线元素为 1，其他元素为 0 的矩阵。

eye(A)：生成大小与矩阵 A 相同的单位阵。

eye()：返回单个 1 值。

(4) 均匀分布随机阵，即各个元素为 0～1 间均匀分布的随机数的矩阵。函数名称为 rand，调用格式如下：

rand(m1,m2)：生成 m1×m2 的随机阵。

rand(m1,m2,…,mn)：生成 m1×m2×…×m*n* 的随机阵。

rand(A)：生成大小与矩阵 A 相同的随机阵。如果 A 是个复数阵，那么 rand(A)也是个复数阵。

rand()：返回一个随机标量。

例 3.7 常用矩阵的生成

```
-->a=zeros(2,4)    //生成全零阵
 a    =
!    0.    0.    0.    0.  !
!    0.    0.    0.    0.  !
-->b=ones(2,4)     //生成全 1 阵
 b    =
!    1.    1.    1.    1.  !
!    1.    1.    1.    1.  !
-->c=eye(2,2)      //生成单位阵
 c    =
!    1.    0.  !
!    0.    1.  !
-->d=rand(2,2)     //生成均匀分布随机阵
 d    =
!     .0124590     .4920584  !
!     .1867539     .7489608  !
-->e=[]     //生成空集矩阵
e    =
[]
```

即 e 将是一个没有任何元素的空矩阵。对变量进行空集操作的目的是在该矩阵元素个数递归增加前对其进行初始化处理。

3.1.4 特殊变量和常数

前面已经看到，当函数没有指定输出时，结果就已放在变量“ans”中。在 SCILAB 中定义了如下一些经常用到的特殊变量和常数：

ans：最近的计算结果，用于自动存储未指定输出变量的表达式的结果。

%eps：浮点运算的相对精度是 SCILAB 计算的误差。

%nan：表示一个无效的数值。

%inf：表示无穷大，可以用来避免浮点溢出。

```
-->%inf-%inf
 ans  =
  - nan
-->10^400
 ans  =
   Inf
```

%i：虚数单位。

```
-->sqrt(-1)
 ans  =
     i
```

%T，%t，%F，%f：布尔常量(真与假)。

%pi：圆周律 3.1415926535897…

```
-->cos(%pi)
 ans  =
  - 1.
```

3.2 基本数值运算函数

矩阵的数值运算操作根据线性代数运算法则定义，按照元素的顺序逐个执行。

3.2.1 加法和减法

由于加减运算是将两个矩阵的对应元素逐个相加减，因此要求参与运算的两个矩阵大小相同，用法如下：

C=A+B
C=A-B

特殊情况是其中一个为标量，那么把另一矩阵的每一个元素与该标量相加减。另外“-”也可作为字符串连接符。如果把“+”和“-”作一元运算符使用，则-A表示将矩阵A中的每个元素取反。

例3.8 加法和减法运算

```
-->A=[1 2 3;4 5 6];
-->B=[1 1 1;2 2 2];
-->C=A+B    //两个矩阵相加
 C    =
!    2.    3.    4.  !
!    6.    7.    8.  !
-->C=C-2    //矩阵与标量相减
 C    =
!    0.    1.    2.  !
!    4.    5.    6.  !
-->s='abc'+'edfgh'    //"+"用于字符串连接
 s    =
abcedfgh
```

3.2.2 乘法运算

乘法运算有两种：一种是根据线性代数定义的两个矩阵的相乘，称为矩阵乘，要求第一个矩阵的列数必须等于第二个矩阵的行数；另一种是两个数组对应元素的相乘，称为数组乘，这种情况下两个矩阵必须有相同的行数与列数。

相应的矩阵乘的计算公式为

$$C_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}$$

数组乘的计算公式为

$$C_{ij} = A_{ik}B_{kj}$$

其运算表达式分别记为

矩阵乘：C=A * B

数组乘：C=A. * B

例 3.9 矩阵乘和数组乘

```
-->A=[1 1 2;2 1 2;1 2 1];
-->B=2 * ones(3,3);
-->C=A * B    //A 和 B 的矩阵乘
 C    =
!    8.    8.    8. !
!   10.   10.   10. !
!    4.    4.    4. !
-->C=A. * B    //A 和 B 的数组乘
 C    =
!    2.    2.    4. !
!    4.    2.    4. !
!    2.    4.    2. !
```

3.2.3 矩阵求逆与除法运算

1. 矩阵求逆

矩阵求逆的运算只适合方阵，即行列数相等的矩阵。矩阵求逆的命令为

B=inv(A)

例 3.10 矩阵求逆

```
-->A=[1 2 3;0 1 0;3 2 1];
-->B=inv(A)
 B    =
```

```
!  -    .125    -    .5        .375  !
!     0.          1.        0.       !
!     .375    -    .5    -   .125  !
```

当矩阵是非满秩时，计算将显示出错信息："奇异矩阵"，例如：

```
-->inv(ones(3,3))
                    !  --error     19
singular matrix
```

2. 矩阵除法

在 SCILAB 中定义了两种矩阵除法：矩阵左除与矩阵右除。矩阵除法在线性代数中是没有定义的，只有矩阵逆的定义。矩阵除法的命令为

矩阵左除：**C=A\B**

矩阵右除：**C=A/B**

如果 A 是一个方阵，那么矩阵左除(A\B)相当于 A 的逆阵左乘 B，即 $A^{-1}B$。这就相当于求方程 AX=B 的解。因此如果 A 是一个 $n\times n$ 的矩阵，B 是个 n 维列向量或 $n\times m$ 的矩阵，矩阵左乘 A\B 返回的就是 AX=B 的解。如果 A 是一个 $m\times n$ 的矩阵，其中 n 近似等于 m，B 是个 m 维列向量或 $m\times n$ 的矩阵，那么 X=A\B 是不定或超定方程的最小二乘解。矩阵右除 A/B 相当于 A 的逆阵左乘 B，即 $B\,A^{-1}$。

例 3.11 矩阵除法

```
A=[1 2 3;0 1 0;3 2 1];
B=[1 1 1]';
-->X=A\B      //矩阵左除
 X   =
!  -    .25  !
```

```
!    1.    !
! -  .25 !
-->B=B'
 B  =
!    1.    1.    1. !
-->X=B/A    //矩阵右除
 X  =
!     .25    0.     .25 !
```

3. 数组除法

数组除法也分两种:数组左除与数组右除。数组除法的命令为

数组左除:C=A.\B

数组右除:C=A./B

数组除法是数组的对应元素相除,因此两个矩阵必须有相同的大小,除非其中一个为标量。其中数组左除是数组 B 和 A 的对应元素相除,即 B_{ij}/A_{ij},数组右除是数组 A 和 B 的对应元素相除,即 A_{ij}/B_{ij} 相除。

例 3.12 数组除法

```
-->A=[2 2 2; 6 6 6];
-->B=[4 4 4;3 3 3];
-->A.\B   //数组左除
 ans  =
!    2.     2.     2.   !
!     .5     .5     .5  !
-->A./B //数组右除
 ans  =
!     .5     .5     .5  !
!    2.     2.     2.   !
-->B./2    //标量运算
```

```
 ans  =
!    2.    2.    2.  !
!    1.5   1.5   1.5 !
```

3.2.4 乘方运算

乘方运算也分两种：矩阵的乘方运算和数组的乘方运算。

矩阵的乘方运算：C=A^p

数组的乘方运算：C=A.^B

1. 矩阵的乘方运算

矩阵的乘方运算要求 A 是一个方阵，p 是一个标量，如果 p 是个正整数，A^p 是矩阵自乘 p 次；如果 p 是个负整数，A^p 是矩阵的逆自乘|p|次。如果 p 不是一个整数，则 A^p=V * D.^p/V，其中，V 为 A 的特征向量矩阵；D 为 A 的特征值矩阵；D.^p 为数组的乘方。

例 3.13 矩阵的乘方运算

```
-->A=[1 2 3;2 1 2;1 3 1];
-->A^2          //相当于 A×A
 ans  =
!    8.    13.   10.  !
!    6.    11.   10.  !
!    8.    8.    10.  !
-->A^(-1)     //结果相当于矩阵求逆
 ans  =
! -  .5     .7     .1  !
!    0.   -  .2     .4  !
!     .5  -  .1   -  .3  !
-->A=[1 0 0;0 2 0; 0 0 3];
-->A^0.5
 ans  =
```

```
!    1.    0.          0.          !
!    0.    1.4142136   0.          !
!    0.    0.          1.7320508   !
```

2. 数组的乘方运算

数组的乘方 C=A.^B 就是矩阵 A 和 B 的对应元素的乘方，即 $A_{ij}{}^{B_{ij}}$。A 和 B 必须有相同大小，除非其中一个为标量，这时计算结果仍为矩阵。如果 p 是标量，A 是矩阵，p.^A 的每一个元素为 $p^{A_{ij}}$，A.^p 的每一个元素为 $A_{ij}{}^{p}$。

例 3.14 矩阵的乘方运算

```
-->A=[1 2 3;2 2 2];
-->B=[2 2 2; 1 2 3];
-->A.^B
 ans    =
!    1.    4.    9.  !
!    2.    4.    8.  !
-->2.^A
 ans    =
!    2.    4.    8.  !
!    4.    4.    4.  !
```

3.2.5 矩阵转置

矩阵转置是将一个大小为 n×m 的矩阵转换为 m×n 的矩阵，运算符为单引号“'”。

例 3.15 矩阵转置

```
-->a=[1 2 3;4 5 6]
 a    =
!    1.    2.    3.  !
!    4.    5.    6.  !
```

```
-->b=a'
 b    =
!    1.    4.  !
!    2.    5.  !
!    3.    6.  !
```

3.3 与矩阵有关的函数

3.3.1 矩阵重新定维

1. matrix 函数

用于改变一个向量或数组的维数或大小，但是元素个数保持不变。

函数格式：y=matrix(v,n,m)。

函数功能：将 v 转为 $n\times m$ 的矩阵。注意这种转化是按列进行的，即先取原矩阵第一列的数据依次排入新尺寸下矩阵中的第一列数据，如若未排满，依次取原矩阵的下一列数据。所以重新定维前后的矩阵元素个数必须相等，否则出错。

例 3.16 将一个 2×3 的矩阵变换为 3×2 的矩阵。

```
-->a=[1 2 3;4 5 6]
 a    =
!    1.    2.    3.  !
!    4.    5.    6.  !
-->b=matrix(a,3,2)
 b    =
!    1.    5.  !
!    4.    3.  !
!    2.    6.  !
```

2. size 函数

用于求矩阵或向量的大小。

函数格式：y=size(x [,sel])。

函数功能：返回矩阵或向量的每一维的大小，结果 y 为一个行向量。如果矩阵 x 是个二维矩阵，则 y 中元素分别代表了该矩阵的行与列的个数。可选项 sel 是个不大于矩阵维数的整数，表示专门对第 sel 维求大小。

例 3.17 求矩阵大小

```
-->a=[1 2 3;4 5 6]
 a    =
!    1.    2.    3.  !
!    4.    5.    6.  !
-->s=size(a)
 s    =
!    2.    3.  !
-->s=size(a,1)      // 求矩阵的行数
 s    =
    2.
```

3.3.2 矩阵提取

1. tril 函数

用于提取下三角矩阵。

函数格式：y=tril(x [,k])。

函数功能：提取矩阵 x 主对角线以下的元素，其他值为 0。可选项 k 为一个整数，如果 k>0 则在主对角线以上扩大 k 级提取范围；如果 k<0 则在主对角线以下缩小 k 级提取范围。k 的默认值为 0。

2. triu 函数

用于提取上三角矩阵。

函数格式：y=triu(x [,k])。

函数功能：提取矩阵 x 主对角线以上的元素，其他值为 0。可选项 k 为一个整数，如果 k>0 则往主对角线以上缩小 k 级提取范围；如果 k<0 则在主对角线以下扩大 k 级提取范围。k 的默认值为 0。

例 3.18 提取矩阵的上下三角矩阵

```
-->tril(a)
 ans   =
!    1.    0.    0.  !
!    4.    5.    0.  !
!    7.    8.    9.  !
-->tril(a,-1)
 ans   =
!    0.    0.    0.  !
!    4.    0.    0.  !
!    7.    8.    0.  !
-->triu(a)
 ans   =
!    1.    2.    3.  !
!    0.    5.    6.  !
!    0.    0.    9.  !
-->triu(a,2)
 ans   =

!    0.    0.    3.  !
!    0.    0.    0.  !
!    0.    0.    0.  !
```

3. diag 函数

产生对角矩阵或提取矩阵的对角线元素。

函数格式:[y]=diag(vm [,k])。

函数功能:

(1) 如果 vm 是一个向量,则产生一个以 vm 的值为对角元素,其他元素的值为 0 的矩阵。参数 k 为可选项,如果 k>0,则 vm 为矩阵主对角线之上的第 k 条对角线;如果k<0,则 vm 为矩阵主对角线之下的第 k 条对角线。

(2) 如果 vm 是一个矩阵,则提取矩阵的对角元素,产生一个列向量。参数 k 为可选项,如果 k>0,则提取主对角线以上第 k 级的对角线元素;如果 k<0,则提取主对角线以下第 k 级对角线的元素。

例 3.19 产生对角矩阵或提取矩阵的对角线元素

```
-->v=[1 2 3];
-->diag(v)    //由向量 v 产生对角阵
 ans    =
!    1.    0.    0.  !
!    0.    2.    0.  !
!    0.    0.    3.  !
-->diag(v,1)  // 由向量产生对角阵,但 v 为主对角线以上的第一级对
              // 角线
 ans    =
!    0.    1.    0.    0.  !
!    0.    0.    2.    0.  !
!    0.    0.    0.    3.  !
!    0.    0.    0.    0.  !
-->a=[1 2 3;4 5 6;7 8 9]
 a    =
!    1.    2.    3.  !
```

```
!    4.    5.    6. !
!    7.    8.    9. !
-->diag(a)  //从矩阵 a 提取对角线
 ans  =
!    1. !
!    5. !
!    9. !
-->diag(a,-2) //提取矩阵 a 的主对角线以下第 2 级对角线
 ans  =
    7.
```

3.3.3 特征值和特征向量

对于一个 $n\times n$ 的方阵 A，如果满足以下关系式：

$$Av=\lambda v$$

则 λ 为 A 的特征值，v 为对应于该特征值 λ 的特征向量。SCILAB 提供求解矩阵特征值和特征向量的函数，分别为 spec 和 bdiag。

1. 求矩阵特征值

函数格式：**evals=spec(A)**

其中：

A：实数或复数方阵。

evals：返回的特征值是个实数或复数向量。

2. 求矩阵特征向量函数

函数格式：**[Ab [,X [,bs]]]=bdiag(A)**

其中：

A：实数或复数方阵；

Ab：实数或复数方阵为分块对角矩阵，如果每一个子块大小为 1，对角线上的元素为矩阵特征值；

X：实数或复数非奇异矩阵，每一列对应于一个特征向量；

bs：整数向量给出各个子块的大小，缺省各元素值为 1。

该函数其实是用来实现矩阵的分块对角化的。对角化的同时可以给出特征向量。

例 3.20 求矩阵特征值和特征向量

```
-->A=[1 2 3;2 3 4;3 4 5];
-->ev=spec(A)    //用 spec 函数求 A 的特征值，结果从小到大排列
 ev    =
!  -   .6234754   !
!      1.152E-17  !
!      9.6234754  !
-->[Ab,X,bs]=bdiag(A)
//返回 A 的对角化后的矩阵 Ab，特征向量矩阵 X 以及各子块的大小。
 bs    =
!      1.  !
!      1.  !
!      1.  !
 X     =
!      .3850898   -   .8276709   -   .4082483   !
!      .5595102   -   .1424137       .8164966   !
!      .7339306       .5428436   -   .4082483   !
 Ab    =
!    9.6234754     0.            0.             !
!    0.          -   .6234754    0.             !
!    0.            0.          -5.845E-16       !
```

3.4 稀疏矩阵

SCILAB 可以处理稀疏矩阵，即大多数元素为 0 的矩阵。其算法就是不存储、不操作那些“0”元素，从而达到节省内存与运算时间的目的。因此稀疏矩阵的计算复杂性取决于非零元素的数

目，与总的元素个数无关。而常规矩阵是全矩阵，将所有的元素都保存在内存中。

3.4.1 创建稀疏矩阵

1. 由全矩阵转化为稀疏矩阵

SCILAB 提供将一个全元素存储矩阵转化为稀疏矩阵的函数 sparse。其简略的调用格式为

S=sparse(X)

其中，X 为全矩阵；S 是稀疏矩阵。如果 X 已经是稀疏矩阵，那么函数调用不产生什么影响。但是如果 X 是一个全矩阵，那么函数将把其中的零元素挤掉，只保留那些非零元素。因此当矩阵大而稀疏时，用稀疏矩阵可以有效地节省存储空间。

例 3.21 用 sparse 函数创建稀疏矩阵

```
-->X=[2 0 1;0 1 0;0 0 3]
 X   =
!    2.    0.    1.  !
!    0.    1.    0.  !
!    0.    0.    3.  !
-->S=sparse(X)
 S   =
(    3,    3) sparse matrix
(    1,    1)          2.
(    1,    3)          1.
(    2,    2)          1.
(    3,    3)          3.
```

注意，把 S 作为稀疏矩阵首先要报告该矩阵的大小，然后是实际存储的那些非零元素及其所在的行列位置。其逆向的操作，即由稀疏矩阵创建全矩阵的操作是由 full 函数完成的，其调用格

式为

X=full(S)

例 3.22 由稀疏矩阵创建全矩阵

```
-->X1=full(S)
 X1    =
!    2.    0.    1. !
!    0.    1.    0. !
!    0.    0.    3. !
```

转换后可以直观地检查矩阵存储的非"0"元素和对应位置。

2. 直接创建稀疏矩阵

这种方法仍是调用 sparse 函数，但是参数不同，其调用格式为

S=sparse(index, values)

如果创建有 N 个元素的稀疏矩阵，则 index 为 $N\times 2$ 的矩阵，index(i,1)与 index(i,2)分别存储了第 i 个元素的行列位置。由 index 存储的最大值决定了该矩阵的大小。values 为 $N\times 1$ 的向量，是稀疏矩阵元素的值。

例 3.23 直接创建稀疏矩阵

```
-->row=[1 1 2 3 3 4 ];          //稀疏矩阵的行号，最大行号为 4
-->col=[3 2 8 1 5 7];           //稀疏矩阵的列号，最大列号为 8
-->index=[row' col']            //合并成下标矩阵
 index    =
!    1.    3. !
!    1.    2. !
!    2.    8. !
!    3.    1. !
!    3.    5. !
!    4.    7. !
```

```
-->val=[1 2 3 4 5 6];                    //6 个元素的值
-->A=sparse(index,val)                   //生成稀疏矩阵
 A    =
(    4,    8) sparse matrix
(    1,    2)          2.
(    1,    3)          1.
(    2,    8)          3.
(    3,    1)          4.
(    3,    5)          5.
(    4,    7)          6.
```

默认情况下产生的矩阵大小由给出的最大行号与列号决定，但是也可以指定矩阵大小。其调用格式为

```
S=sparse(index,values,dim)
```

dim 为 1×2 的向量，它给出矩阵的大小。例如：

```
-->A=sparse(index,val,[8,8])
 A    =
(    8,    8) sparse matrix
(    1,    2)          2.
(    1,    3)          1.
(    2,    8)          3.
(    3,    1)          4.
(    3,    5)          5.
(    4,    7)          6.
```

用 Size 函数操作可以返回稀疏矩阵的大小。

3. 特殊稀疏矩阵

(1) 非零元素都为 1 的矩阵

通过 spones 函数，可以根据一个已有的稀疏矩阵创建一个结构相同但元素全为 1 的稀疏矩阵，其调用格式为

S1＝spones(S)

例 3.24 创建非零元素都为 1 的矩阵

```
-->A=[0 2 0 0;4 0 1 0;0 0 3 0;0 5 0 3];
-->S=sparse(A);
-->S1=spones(S);
-->full(S1)
 ans   =
!    0.    1.    0.    0. !
!    1.    0.    1.    0. !
!    0.    0.    1.    0. !
!    0.    1.    0.    1. !
```

(2) 稀疏单位阵

稀疏单位阵即对角元素为 1，其他元素为 0 的单位阵。它可以生成与给定矩阵相同大小的稀疏单位阵，也可以通过输入行列值指定稀疏单位阵的大小。

下面给出利用上面的 S 创建 4×4 的稀疏单位阵的例题：

例 3.25 创建稀疏单位阵

```
-->SI=speye(S)
 SI   =
(    4,    4) sparse matrix
(    1,    1)          1.
(    2,    2)          1.
(    3,    3)          1.
(    4,    4)          1.
```

或者直接指定稀疏单位阵的大小：

```
-->SI=speye(3,4);
-->full(SI)   //显示全矩阵
 ans   =
!    1.    0.    0.    0. !
```

```
!    0.    1.    0.    0.  !
!    0.    0.    1.    0.  !
```

(3) 稀疏随机阵

稀疏随机阵为非零元素是0～1之间随机数的矩阵，而且随机数的分布由指定的概率值确定。调用格式为

```
S=sprand(row,col,pf [,type])
```

其中最后一个可选参数 type 的取值为"uniform"或"normal"，即指定所产生的随机数是服从均匀分布还是正态分布。缺省情况下为均匀分布。

例 3.26 创建稀疏随机阵

```
-->S=sprand(4,4,0.2);
-->full(S)
 ans    =
!      .4148104      .2806498      0.      0.  !
!      0.            0.            0.      0.  !
!      0.            0.            0.      0.  !
!      0.            .1280058      0.      0.  !
```

(4) 稀疏全零阵

稀疏全零阵的所有元素都为零，实际上不占用内存，但是可以用来定义一个稀疏矩阵。创建函数为 spzeros，用法与 speyes 一样。设 S 与上例相同：

例 3.27 创建稀疏全零阵

```
-->spzeros(S)
 ans    =
(4,     4) zero sparse matrix
相当于
-->S0=spzeros(4,4);
-->full(S0)
```

```
 ans    =
!    0.    0.    0.    0.  !
!    0.    0.    0.    0.  !
!    0.    0.    0.    0.  !
!    0.    0.    0.    0.  !
```

3.4.2 稀疏矩阵的查看

1. 稀疏矩阵的查看

对于稀疏矩阵一个基本的要求是查看非零元素的信息。在SCILAB中用 spget 函数实现。其调用格式为

```
[index,values,dim]=spget(A)
```

其中，index 为非零元素的行列信息；values 为非零元素的值；dim 为矩阵的大小。通过调用 spget 函数，这些信息都得到。

例 3.28 查看稀疏矩阵中非零元素信息

```
-->A=[0 2 0 0;4 0 1 0;0 0 3 0;0 5 0 3];
-->S=sparse(A);
-->[index,values,dim]=spget(S)
 dim    =
!    4.    4.  !
 values    =
!    2.  !
!    4.  !
!    1.  !
!    3.  !
!    5.  !
!    3.  !
 index    =
!    1.    2.  !
!    2.    1.  !
```

```
!    2.    3. !
!    3.    3. !
!    4.    2. !
!    4.    4. !
```

find 函数可以返回非零元素的下标。

```
-->[i,j]=find(S)
 j    =
!    2.    1.    3.    3.    2.    4. !
 i    =
!    1.    2.    2.    3.    4.    4. !
```

2. 稀疏矩阵可视化

利用 SCILAB 的图形，可以将稀疏矩阵中非零元素的分布以图形方式直观地显示出来。

例 3.29 稀疏矩阵的可视化

```
S=sprand(80,100,0.2,'normal');          //生成约 20%的非零元素的随
                                        //机稀疏矩阵
[index,values,dim]=spget(S);
x=index(:,1);y=index(:,2);
plot2d(x,y,-2)                          //非零元素由"×"符标示(见
                                        //图 3.1)
```

3. 稀疏矩阵的计算

稀疏矩阵的计算与常规全矩阵计算方式基本上是一样的。比如矩阵加减法，求矩阵大小等。对于一元函数，如果输入为稀疏矩阵，则输出一般也是稀疏矩阵，例如 diag(S)；对于二元运算符，如果两个操作数都是稀疏的，则运算结果也是稀疏的；如果两个操作数一个是稀疏的，另一个是全元素的，除非运算保留稀疏性，否则给出全元素结果。比如 S+A，S * A，S/A 的结果为全矩阵，而 S. * A的结果为稀疏的。稀疏矩阵的计算复杂度与矩阵的非零元

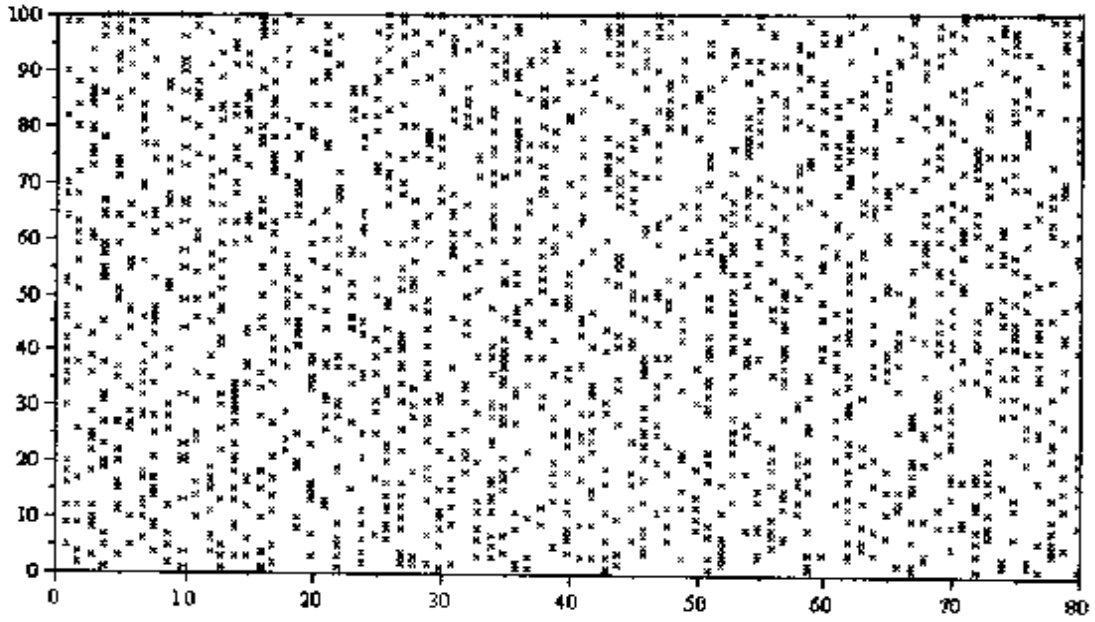


图 3.1 稀疏矩阵的可视化

素个数成正比，但是与矩阵总的元素个数无关。

例 3.30 稀疏矩阵的计算

```
-->S=speye(4,4);                                    //定义稀疏单位阵 S
-->A=[1 2 5 3; 6 3 3 3; 5 2 0 9; 2 3 5 3];          //定义全矩阵 A
-->S(2,4)=1;
-->S+A
 ans    =
!    2.    2.    5.    3.  !
!    6.    4.    3.    4.  !
!    5.    2.    1.    9.  !
!    2.    3.    5.    4.  !
-->S. * A
 ans    =
(    4,    4) sparse matrix
(    1,    1)          1.
(    2,    2)          3.
(    2,    4)          3.
```

```
(    3,    3)        0.
(    4,    4)        3.
```

SCILAB还提供了其他专门用于稀疏矩阵的函数，这里就不一一介绍了。

3.5 数据统计

1. 求向量或矩阵中的最大值(max)

调用格式如下：

[m]=max(A) 求向量或矩阵A中所有元素的最大值，并将结果返回到m中。

[m ,i]=max(A) m返回A的最大值，i保存最大值的下标。

[m ,i]=max(A, 'r') m返回一个行向量，其中的元素对应于A的每一列的最大值。

[m ,i]=max(A, 'c') m返回一个列向量，其中的元素对应于A的每一行的最大值。

[m [,i]]= max(A1,A2,…,An) m的大小与Aj相同，各个参数Aj为相同大小的矩阵或标量，m的每一个元素为所有Aj对应位置上的最大值。i为取得最大值的A的序号j。

例 3.31 求最大值示例

```
A    =
!    3.    1.    1.  !
!    1.    3.    1.  !
!    1.    1.    5.  !
-->m=max(A)                    //返回A的最大元素的值
 m    =
    5.
```

```
-->[m,i]=max(A)          //返回 A 的最大元素的值及相应位置
 i    =
!    3.    3.  !
 m    =
    5.
--> A=[1 5 2;4 3 5;6 7 1];
 A    =
!    1.    5.    2.  !
!    4.    3.    5.  !
!    6.    7.    1.  !
-->[m,i]=max(A,'r')      //返回 A 的每列最大元素的值及相应行号
 i    =
!    3.    3.    2.  !
 m    =
!    6.    7.    5.  !
[m,i]=max(A,'c')         //返回 A 的每行最大元素的值及相应列号
 i    =
!    2.  !
!    3.  !
!    2.  !
 m    =
!    5.  !
!    5.  !
!    7.  !
-->B=3 * ones(3,3)
 B    =
!    3.    3.    3.  !
!    3.    3.    3.  !
!    3.    3.    3.  !
-->m=max(A,B)            //求 A 和 B 的最大值矩阵
 m    =
!    3.    5.    3.  !
```

```
!    4.    3.    5. !
!    6.    7.    3. !
 i  =
!    2.    1.    2. !
!    1.    1.    1. !
!    1.    1.    2. !
```

求向量或矩阵中的最小值(min)的调用格式与 max 相同。

2. 求向量或矩阵的均值(mean)

调用格式如下：

y=mean(x) 求向量或矩阵 x 中所有元素的平均值，并将结果返回到 y 中。

y=mean(x,'r') 返回一个行向量 y，该向量的每一个元素为 x 中对应列的平均值。

y=mean(x,'c') 返回一个列向量 y，该向量的每一个元素为 x 中对应行的平均值。

例 3.32 求平均值示例

```
-->x=[1 3 2; 2 4 6; 3 7 2]
 x  =
!    1.    3.    2. !
!    2.    4.    6. !
!    3.    7.    2. !
-->y=mean(x)                    //求 x 所有元素的平均值
 y  =
    3.3333333
-->y=mean(x,'r')                //按列求 x 的平均值
 y  =
!    2.    4.6666667    3.3333333 !
-->y=mean(x,'c')                //按行求 x 的平均值
 y  =
```

```
!    2.    !
!    4.    !
!    4.    !
```

3. 求元素之和(sum)

调用格式如下：

y=sum(x) 求向量或矩阵 x 中所有元素的和，并将结果返回到 y 中。

y=sum(x,dim) 求矩阵 x 的第 dim 维中所有元素的和，并将结果返回到 y 中。

例 3.33 求和示例

```
-->x=[1 2 3;4 5 6;7 8 9]
 x    =
!    1.    2.    3.    !
!    4.    5.    6.    !
!    7.    8.    9.    !
-->y=sum(x)                    //求所有元素的和
 y    =
    45.
-->y=sum(x,1)                  //求每一列的和
 y    =
!    12.    15.    18.    !
-->y=sum(x,2)                  //求每一行的和
 y    =
!    6.    !
!    15.   !
!    24.   !
```

4. 求累计和(cumsum)

调用格式如下：

y=cumsum(x) 求向量或矩阵 x 按列的方向累计的和，并将

结果返回到 y 中。y 与 x 同维。

例 3.34 求累计和示例

```
y=cumsum(x,'r')   在 y 的每一行保存列方向的累计和。
y=cumsum(x,'c')   在 y 的每一列保存行方向的累计和。
-->x=[1 2 3;4 5 6;7 8 9]
 x  =
!   1.    2.    3.  !
!   4.    5.    6.  !
!   7.    8.    9.  !
-->y=cumsum(x)              //按列顺序所有元素累计求和
 y  =
!   1.    14.   30.  !
!   5.    19.   36.  !
!   12.   27.   45.  !
-->y=cumsum(x,'r')          //每列累计求和
      y  =
!   1.    2.    3.   !
!   5.    7.    9.   !
!   12.   15.   18.  !
-->y=cumsum(x,'c')          //每行累计求和
 y  =
!   1.    3.    6.   !
!   4.    9.    15.  !
!   7.    15.   24.  !
```

5. 求矩阵元素的积(prod)

调用格式如下：

y=prod(x) 求向量或矩阵 x 中所有元素的积，并将结果返回到 y 中。

y=prod(x,'r') 返回一个行向量 y，其中的元素对应于 A 的每一列的乘积。

y=prod(x,'c') 返回一个列向量 y,其中的元素对应于 A 的每一行的乘积。

例 3.35 矩阵元素求积示例

```
-->x=[1 2 3;2 2 1;3 1 1]
 x    =
!    1.    2.    3.  !
!    2.    2.    1.  !
!    3.    1.    1.  !
-->y=prod(x)                    //所有元素乘积
 y    =
    72.
-->y=prod(x,1)                  //按列求积
 y    =
!    6.    4.    3.  !
-->y=prod(x,2)                  //按行求积
 y    =
!    6.  !
!    4.  !
!    3.  !
```

6. 求累计积(cumprod)

调用格式如下:

y= cumprod (x) 求向量或矩阵 x 按列的方向累计积,并将结果返回到 y 中。y 与 x 同维。

y= cumprod (x,'r') 在 y 的每一行保存列方向的累计积。

y= cumprod (x,'c') 在 y 的每一列保存行方向的累计积。

例 3.36 矩阵元素求累计积示例

```
-->y=cumprod(x)
 y    =
!    1.    12.    72.  !
```

```
!    2.    24.    72.  !
!    6.    24.    72.  !
-->y=cumprod(x,'r')
 y    =
!    1.    2.    3.  !
!    2.    4.    3.  !
!    6.    4.    3.  !
-->y=cumprod(x,'c')
 y    =
!    1.    2.    6.  !
!    2.    4.    4.  !
!    3.    3.    3.  !
```

函数的第二个参数"r"或"c"可以用整数 1 或 2 代替，分别表示行和列。

3.6 数值积分

3.6.1 一重定积分

函数格式：[v,err]= intg (a, b, f [,ea [, er]])

其中：

a，b：积分的上下限，为实数；

f：被积函数名；

ea：积分结果的绝对误差容限，实数，缺省值为 0；

er：积分结果的相对误差容限，实数，缺省值为 1. E-8；

v：函数 intg 的积分结果；

err：结果绝对误差估计值。

intg(a,b,f) 计算函数 f 的自变量由 a 到 b 的定积分值，计算精度满足以下表达式：

abs (I −v)<=max (ea , er * abs (I))

即通过该函数得到的数值积分结果与真实结果的误差在误差允许范围内，其中 I 表示积分的实际精确结果。下面通过一个例子说明该函数的调用方法。

例 3.37 假设函数表达式为 $y=f(x)=x^2$，积分的上下限为[0,2]，求积分结果

```
deff('[y]=f(x)','y=x*x')                    //定义函数 y=x*x
result=intg(0,2,f)                          //调用定积分函数
result    =
    2.6666667
[result,error]=intg(0,2,f)
//如果要返回计算结果误差，则可以如此调用
error    =
    2.961E-14
 result    =
    2.6666667
```

如果用手工计算函数的积分值来验证这一结果，则有

$$\int_0^2 x^2 \mathrm{d}x = \frac{1}{3}x^3\Big|_0^2 = \frac{8}{3}$$

结果一致。

3.6.2 二重定积分

函数格式： [I , err] = int2d (X , Y , f [, params])

其中：

I：函数 int2d 的积分结果。

err：积分误差估计值。

X：一个 3×N 的矩阵，包含 N 个三角形顶点的 X 坐标信息。

Y：一个 3×N 的矩阵，包含 N 个三角形顶点的 X 坐标信息。

f：被积函数名。

params：积分参数为实数向量[tol，iclose，maxtri，mevals，iflag]，缺省值为[1.d-10，1，50，4000，1]，其中，

tol：积分结果的误差容限，如果 iflag=0，则 tol 为相对误差；如果 iflag=1，则 tol 为绝对误差。

iclose：一个整数参数，用于选择 LQM0 或 LQM1 积分方法。如果 iclose=1，则选择 LQM1，如果 iclose 取其他值，则使用 LQM0。LQM0 只使用三角形的内部点，一般 LQM1 比 LQM0 更加精确，但是需要对三角形的边界点做较多的判断。除非三角形的边界上将产生奇异值，通常采用 LQM1 会好一些。

maxtri：区域三角化时三角形的最大个数。

用该函数求二维区域上的积分时，首先将该区域三角化。int2d 计算在一个包含了 N 个三角形的区域内函数 f 的二维积分。计算结果同时返回积分的计算误差，以与计算的允许误差相比较。int2d 在计算每个三角形面积时使用了“局部面积模块(LQM)”，如果总的误差估计大于误差容限，则在绝对误差最大的那个三角形的最大边上增加中线，使其分割成两个三角形。然后在这两个子三角形上应用 LQM 得到新的积分值与误差估计。重复上述过程，直到满足下述条件之一：①误差估计小于误差容限；②产生的三角形个数超出了输入参数 maxtri；③函数探测出舍入误差开始影响结果。

例 3.38 计算函数 z=2 在 0≤x≤1，0≤y≤1 的正方形区域(见图 3.2)内的双重积分值

```
X=[0,0;1,1;1,0];      // X,Y 为两个三角形的坐标
Y=[0,0;0,1;1,1];
deff('z=f(x,y)','z=2')
```

```
[I,e]=int2d(X,Y,f)
```

结果为

```
e    =
    6.661E-16
 I   =
    2.
```

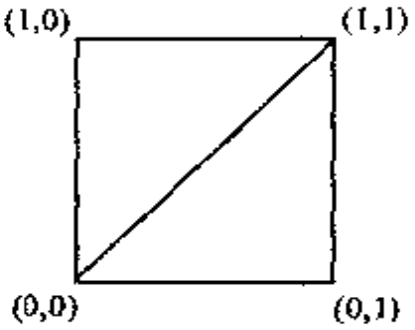


图 3.2 正方形区域

而实际积分结果为该立方体体积等于2，结果一致。

如果被积函数为 z=cos(x+y)，

```
-->[I,e]=int2d(X,Y,f)
 e       =
    3.569E-11
 I       =
    .4967514
```

由于是由平面替代曲面，所以误差有所增加。

函数使用前首先要将区域三角化，这个工作可以由 mesh2d 函数自动完成。此例中直接给出三角化后两个三角形的 X 与 Y 坐标。

3.6.3 三重定积分

有了前面的基础，不难了解三重积分函数的使用。积分单元为四面体。

函数格式：[result,err]=int3d(X,Y,Z,f [,nf[,params]])

其中：

X：一个 $4\times N$ 的实数矩阵，包含 N 个四面体顶点 X 的坐标信息。

Y：一个 $4\times N$ 的实数矩阵，包含 N 个四面体顶点 Y 的坐标

信息。

Z：一个 $4\times N$ 的实数矩阵，包含 N 个四面体顶点 Z 的坐标信息。

f：积分函数

nf：积分函数个数，缺省值为 1。

params：实数向量[**minpts**, **maxpts**, **epsabs**, **epsrel**]. 缺省值为[0, 1000, 0.0, 1.d-5]。其中：

minpts 函数评价最小数量；

maxpts 函数评价最大数量；

epsabs 积分结果的绝对误差容限；

epsrel 函数 int3d 的相对误差容限；

Result：积分结果，若有多个积分函数，则为向量；

err：积分绝对误差估计值。

该函数计算三重积分的近似数值解。积分区域以四面体表示，为得到理想的积分结果 i(k)，要求误差满足下式：

abs(i(k)－result(k))<=max(epsabs,epsrel * abs(i(k))

如果不满足，则将产生误差最大的四面体进行分割，然后计算新的四面体集合下的积分值与误差。重复这个过程，直到精度满足要求，或者已经达到四面体个数的上限(MAXPTS)。

如果 int3d 输入多个函数参与积分，那么积分空间的划分对于不同函数是一样的，因此当函数形式相近时，多个函数的积分求值调用一个 int3d 可以节省时间。

例 3.39 三重积分的计算

```
X=[0;1;0;0];
Y=[0;0;1;0];
Z=[0;0;0;1];
deff('v=f(xyz,numfun)','v=exp(xyz'' * xyz)')
[RESULT,ERROR]=int3d(X,Y,Z,'int3dex')
```

```
ERROR   =
    .0000020
 RESULT   =
    .2278
```

3.7 优化计算

3.7.1 数据拟合

Datafit 为基于测量数据的参数拟合，找到的参数使拟合误差最小，其调用格式为

```
[p,err]= datafit( [imp,] G [,DG], Z [,W], [contr], p0, [algo],
[df0, [mem]], [work], [stop],['in'] )
```

其中：

imp：用于跟踪模式的设定，是一个标量，有以下几种取值情况：

imp=0 除错误信息以外不报告任何结果

imp=1 初始和最终报告

imp=2 每次迭代做报告

imp>2 线性搜索时报告

大部分的报告结果以 SCILAB 标准格式输出。

G：目标函数描述，形如 e =G(p,z)其中，p 为参数向量；z 为测量组。

Z：用于存放测量数据的矩阵，其形式为 matrix [z1,z2,…,zn]，其中 zi (zn ×1)为第 i 个测量结果。

w：权重矩阵。

contr：参数约束，形为'b',binf,bsup,其中，binf 及 bsup 是与

p0 维数相同的实向量，分别是参数 p 的上限与下限。

p0：参数初始的设置值，可根据经验给定。

algo：取值为'qn','gc'或'nd'，分别表示 Quasi-Newton 法(缺省)、共轭梯度法和 Non-differentiable 算法。

stop：用于控制算法收敛性的备选参数，格式为

```
stop='ar',nap,[iter[,epsg[,epsf[,epsx]]]]
```

其中：

ar 停止规则选择的保留关键字

nap 允许调用函数的最大次数

iter 允许的最大迭代次数

epsg 梯度模(gradient norm)的域值

epsf 控制 f 下降的域值

epsx 控制 x 变化的域值

'in'：初始化参数的保留键值。

p：列向量，找到的最优解。

err：最小二乘误差，为一个标量。

datafit 通过寻找模型参数将实验数据与一个已知形式的模型相拟合。其目的是对于一给定的函数形成 G(p,z)，datafit 找出最好的参数向量 p，使得对于各个测量数据 zi，G(p,zi)的值近似为 0。这一过程通过将 G(p,z1)'WG(p,z1)+G(p,z2)'WG(p,z2)+…+G(p,zn)'WG(p,zn)最小化来实现。

例 3.40 数据拟合

```
deff('y=FF(x)','y=a*(x-b)+c*x.*x')      //定义模型的函数形
                                        //式，其中 a,b,c 为
                                        //参数。
X=[];Y=[];
a=34;b=12;c=14;
for x=0:.1:3, Y=[Y,FF(x)+50*(rand()-.5)];X=[X,x];end
```

```
//根据实际参数将值进行扰动,以产生实验数据集 Z。
Z=[Y;X];
deff('e=G(p,z)','a=p(1),b=p(2),c=p(3),y=z(1),x=z(2),e=
y-FF(x)')  // 定义目标函数
[p,err]=datafit(G,Z,[3;5;10])
//设参数的初始值为[3;5;10],开始数据拟合
```

实验的一次执行结果如下：

由于每次执行产生的数据集不同，拟合得到的参数也会不同。

```
 err   =
    6226.1904
 p    =
!     30.590489  !
!     13.128363  !
!     14.779028  !
 xset('window',0); xbasc();
 plot2d(X',Y',-1) ; plot2d(X',FF(X)',5,'002')
 a=p(1),b=p(2),c=p(3);plot2d(X',FF(X)',12,'002')
```

将离散数据、真实模型与拟合模型曲线输出，得到图 3.3 所示的图形。

3.7.2 非线性优化

函数格式：[f,xopt]=optim(costf,x0)

或

[f,[xopt,[gradopt,[work]]]]=optim(costf,[contr],x0,['algo'],[df0,[mem]],[work],[stop],['in'],[imp=iflag])

根据给定的目标函数搜索最佳的函数参数，使目标函数值最小，其中：

costf：目标函数名称。

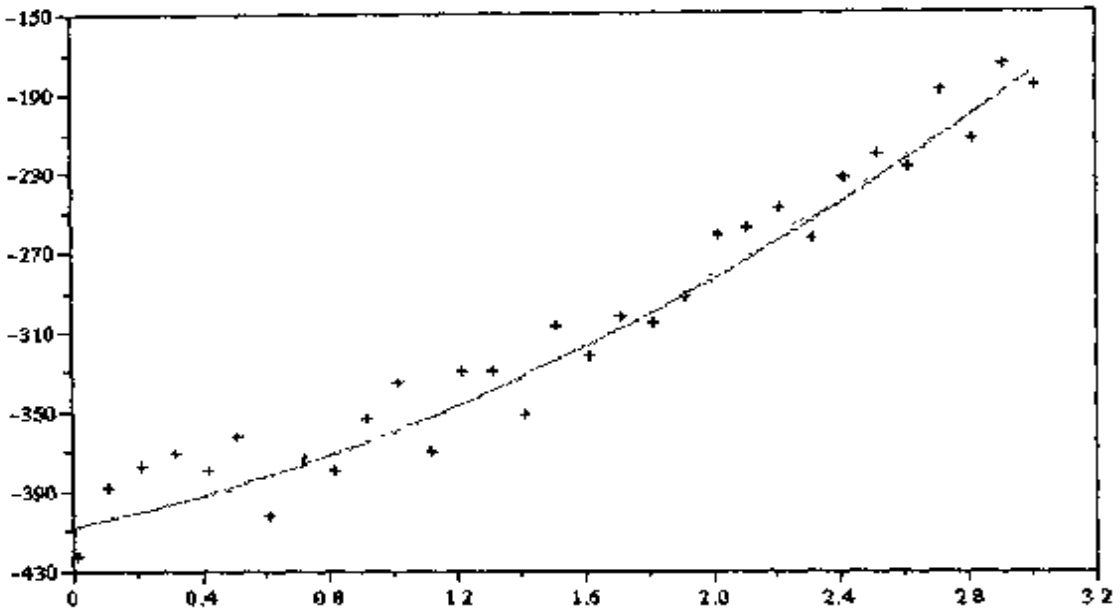


图 3.3 数据拟合曲线

x0：变量初始值，实数向量。

f：最终目标函数值。

xopt：找到的最佳向量。

contr：参数边界'**b**'，**binf** 和 **bsup** 中 **binf**，**bsup** 是与 x0 大小相同的向量，分别表示 x 的下限与上限。

'**algo**'：'**qn**'或'**gc**'或'**nd**' 分别表示 quasi-Newton 法（缺省值），共轭梯度法和 non-differentiable 算法。注意'**nd**'不接受 x 的边界值。

df0：初次迭代下降值的猜测。实标量，缺省值为 1。

mem：整数，用于 Hessian 的变量个数 Hessian(algo=gc 或 nd 时)，缺省值为 6 左右。

stop：控制算法收敛性的一系列可选参数，格式为

stop='ar',nap, [iter [,epsg [,epsf [,epsx]]]]

其中，'ar'为保留关键字，用于程序中止规则的选择。中止规则约束有如下几种：

nap 调用 costf 的最大允许次数。

iter 允许的最大迭代次数。

epsg 梯度向量的模的域值。

epsf 控制 f 下降的域值。

epsx 控制 x 变动的域值。这是个向量或矩阵，可用来放大或缩小 x 的值。

'in'：当 costf 以 Fortran 函数给出时，作为参数初始化的保留关键字。

'imp=iflag'：用于设置跟踪模式。iflag 的取值与含义如下：

iflag=0 除了错误信息外不报告其他信息。

iflag=1 初始与最终信息报告。

iflag=2 每次迭代报告一条信息。

iflag>2 报告线性搜索警告。

这些报告信息大多以 SCILAB 标准格式输出。

gradopt：costf 在 xopt 的梯度。

work：用于 quasi-Newton 法的工作数组，由 optim 函数自动激活，可以用于输入参数加速计算。可以看出，optim 函数与 datafit 函数有很多参数是一致的。

函数说明：用于无约束问题的优化求解。

例 3.41 优化问题

```
xref=[1;2;3];x0=[1;-1;1]
deff('[f,g,ind]=cost(x,ind)','f=0.5*norm(x-xref)^2,g=x-xref');
[f,xopt]=optim(cost,x0)                    //简单的调用形式
xopt    =


!    1.  !
!    2.  !
!    3.  !
```

```
 f    =
0.

 [f,xopt,gopt]=optim(cost,x0,'gc')   // 共轭梯度法
gopt    =
!     0.  !
!     0.  !
!     0.  !
 xopt    =
!     1.  !
!     2.  !
!     3.  !
      f    =
      0.
[f,xopt,gopt]=optim(cost,'b',[-1;0;2],[0.5;1;4],x0)
                          // 设定 x 的边界
gopt    =

!  -     .5    !
!  -     1.    !
!     0.       !
 xopt    =
!        .5    !
!     1.       !
!     3.       !
 f    =
         .625
 [f,xopt,gopt]=optim(cost,'b',[-1;0;2],[0.5;1;4],x0,'gc')
                  //设定 x 的边界并用共轭梯度法
gopt    =
!  -     .5    !
!  -     1.    !
```

```
!      0.     !
 xopt    =
!       .5    !
!      1.     !
!      3.     !
 f    =
         .625
 [f,xopt,gopt]=optim(cost,'b',[-1;0;2],[0.5;1;4],x0,'gc','ar',3)
                    //控制梯度向量的模的域值
gopt    =
! -     .5          !
! -     1.          !
! -     .4933333    !
 xopt    =

!       .5          !
!      1.           !
!      2.5066667    !
 f    =

         .7466889
```

3.8 插值与样条

3.8.1 插值函数

插值就是在原始数据之间按一定关系插入新的数据点，以便分析数据变化规律。插值函数属于 SCILAB 提供的基本函数。

1、Interpln 线性插值函数

调用格式：[f]=interpln(xy,xd)

其中：

xy：xy 平面上的原始点坐标，x 的坐标值由小到大排列；

xd：插值点 x 坐标值，从小到大排列；

f：根据原始数据进行线性标值后对应于 x 的坐标值。

线性插值函数非常简单，仅通过连接相邻的两个数据点就可得到 x 坐标值位于这两个点之间的插值点的坐标。

例 3.42 线性插值函数

```
x=[1 10 20 30 40];                    //原始点 x 坐标
y=[1 30 -10 20 40];                   //原始点 y 坐标
yi=interpln([x;y],-4:45);             //-4:45 为插值点的 x 坐标
plot2d((-4:45)',yi',[3],"000");       //绘制插值结果
```

2. Interp 样条插值函数

调用格式：[f0 [,f1 [,f2 [,f3]]]]=interp(xd,x,y,d)

其中：

xd：实数向量，为指定的插值点的 x 坐标。

x，y，d：3 个实数向量，分别为原始数据点的 x，y 坐标以及其一阶导数，可以由 splin 函数得到，参见 splin 函数的说明。

fi：与插值点相对应的实数向量，其中 f0 为插值点的 y 坐标，f1，f2，f3 分别为插值点上的一阶、二阶、三阶导数。

该函数根据原始数据点的坐标以及该点的一阶导数进行插值，规定插值函数的一阶导数在数据点上与 d 相等，从而使拟合显得光滑。

3.8.2 样条函数

splin 函数，其调用格式为

d=splin(x,f [,"periodic"])

其中：

x：实数向量，表示原始数据点的横坐标，必须单调上升；

f：实数向量，表示原始数据点的纵坐标；

periodic：字符串标识，用于寻找周期性样条函数，这时必须有 f(1)=f(n)；

d：一阶导数向量。

该函数与 interp 配合使用，用来求三次样条。

例 3.43 样条函数(见图 3.4)

```
x=0:0.5:10;f=sin(x);
d=splin(x,f);
S=interp(0:0.1:10,x,f,d);
plot2d(x',f',-1);
plot2d((0:0.1:10)',S',2,'000')
```

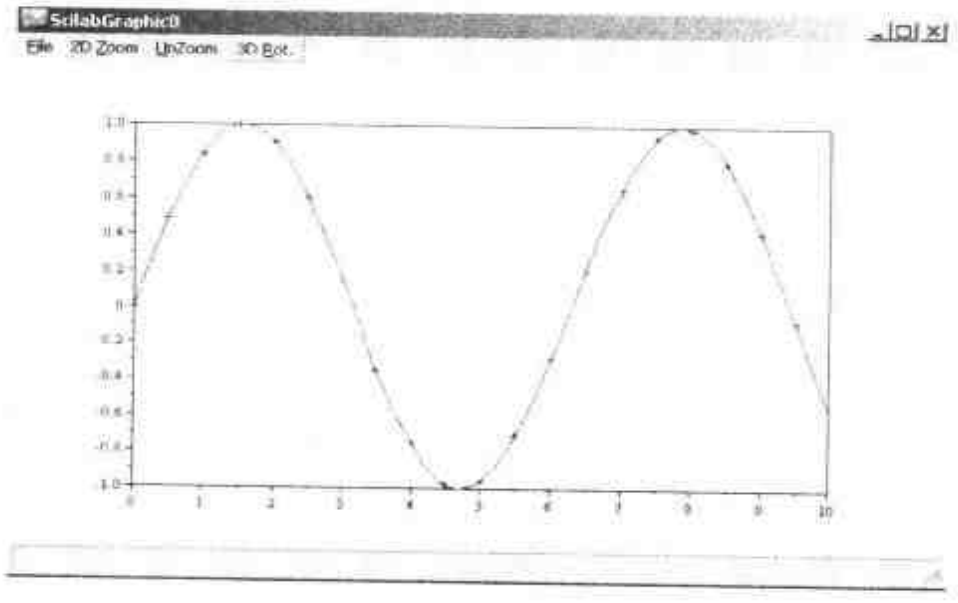


图 3.4 样条函数逼近曲线

这里原始数据点是图中标出的离散点，原始数据的 x 值的间

隔为 0.5，插值函数用间隔 0.1 的数值点进行插值，结果逼近原先的 sin 曲线。

小结：本章介绍了 SCILAB 环境下的常用数值计算。首先介绍了最基本的矩阵生成方法及基本的矩阵运算，例如加，减，乘，除，乘方，矩阵转置，重新定维，提取对角线，求特征值与特征向量；然后介绍了特殊的矩阵——稀疏矩阵，接着介绍了利用矩阵进行的一些简单统计计算。第 5，6 节分别介绍了两种 SCILAB 应用，数值积分和优化拟合。

第4章

SCILAB数据类型及程序设计

SCILAB将全部数据分为3大类型：标量式类型、矩阵式类型及特殊类型。各大类型中又包括各种具体类型。例如标量式类型中包括：数值变量、布尔变量、多项式和字符串。以上述具体类型作为元素的矩阵被定义为矩阵式类型数据，它也可以理解为标量式类型数据的广义表达形式。特殊数据类型包括表(list)和函数类型。在上一章中已经介绍了数值变量类型，本章将全面介绍其他具体数据类型以及基本的程序设计方法。

4.1 数据类型

4.1.1 布尔矩阵

布尔型的数据只有两个值：真或假。一个关系式的运算结果会返回布尔型数据，它常用于条件判断。布尔型矩阵的操作与普通矩阵相同，例如可以相加和转置等。

1. 布尔矩阵的生成

布尔常量是%t和%f，输出在屏幕上为“T”和“F”。元素类型

全为布尔类型的矩阵为布尔矩阵。这里顺便介绍以下 SCILAB 中的关系运算符：

<	小于	<=	小于等于
>	大于	>=	大于等于
==	等于	<>	不等于

由于关系运算符的优先级别比赋值号“=”要高，所以可以直接给一个关系表达式赋给一个布尔变量，例如 a=c>b，a 为布尔变量。两个相同类型矩阵进行关系运算，生成一个布尔矩阵。也可以通过直接赋值得到。

例 4.1 布尔型数据的生成

```
-->%t
%t =
T
-->a=[1,2]>=[1,3]
 a  =
! T F !
-->[1,2]==1   //各个矩阵元素与一个标量比较
ans =
! T F !
-->a=1:5
 a   =
!    1.    2.    3.    4.    5. !
-->b=a(a>2)
 b   =
!    3.    4.    5. !
```

2. 布尔矩阵的运算

和其他语言一样，SCILAB 有基本的逻辑运算功能。逻辑运算符有

&　　逻辑与

| 逻辑或

~ 逻辑非

例 4.2 布尔型矩阵的逻辑运算

```
-->A=[%t,%f,%t,%f,%f,%f]
 A   =
! T F T F F F !
-->B=[%t,%f,%f,%t,%f,%t]
 B   =
! T F F T F T !
-->A & B        //与运算
 ans   =
! T F F F F F !
-->A | B        //或运算
 ans   =
! T F T T F T !
-->~A           //非运算
 ans   =
! F T F T T T !
```

4.1.2 字符串矩阵

字符在输入输出信息显示中具有重要作用，几乎是每一种编程语言不可缺少的功能。SCILAB 提供了字符处理的功能。

在 SCILAB 中，字符串用单引号或双引号将字符括起来表示，例如'abcdef'或"abcdef"。字符串可以作为矩阵或向量的元素，构成字符串矩阵或向量。下面介绍 SCILAB 中几种常用的操作字符串的方法。

1. 字符串的产生

无论是单个字符串还是字符串矩阵或向量都可以通过直接赋值产生。矩阵的赋值方法与前面介绍的相同。

例 4.3 字符串赋值

```
-)s="abcdef"                                    //单个字符串直接赋值
 s    =
 abcdef
-)sm= ['this','is'; 'a 2x2','matrix']          //产生字符串矩阵
 sm    =
! this    is          !
!                     !
! a 2x2    matrix     !
```

获得字符串的间接方法是通过函数间接转换得到，选择哪一种方法可根据具体使用场合而定。可以实现这一目的的函数有

code2str：将整型 SCILAB 代码转换为相应的字符，问题是这种转换后的编码不是一般的 ASCII 编码。与 code2str 功能相反的函数为 str2code。

string：将矩阵转换为字符类型。

emptystr：产生空字符串

例 4.4 通过函数得到字符串

```
-)s= code2str([-28 12 18 21 10 11]) //通过代码得到相应字符
 s    =
scilab
-)s=string(rand(1,2))                          //将实数向量转为字符串向量
 s    =
!.0002211     .3303271    !
```

在最后一个例子中虽然结果看起来还是一个实型向量，但实际上是字符型向量，在显示时与实型或整型数据不同的是字符型

数据按实际宽度显示。比较下面两种变量的默认显示：

```
-->a='1234.56'          //字符串
 a    =
 1234.56
-->a=1234.56            //实型数据
 a    =
    1234.56
--> s=emptystr(2,3)     //产生空字符串
 s    =
!        !
!        !
!        !
```

2. 字符串连接

通过字符串连接可以将短的字符串组合成长的字符串。如前所述，用加号＋可以实现这一功能。当参与运算的是两个大小相同的字符串数组时，对应位置的字符串元素相连接，产生一个新的字符串数组。与字符串连接有关的运算符与函数如下：

＋：将两个字符串左右相连。如果参与运算的是两个大小相同的字符串矩阵，则将对应位置上的字符串连接生成一个新的字符串数组。

函数strcat：调用格式为 **txt=strcat(vstr [,strp])**

函数功能为将一个字符串数组 vstr 中的各个元素连接起来形成单个字符串 txt。在 strcat 函数中增加第二个可选字符串类型参数 strp 可以在各个元素间插入一个字符串，以起到隔离数据的作用。

例 4.5 字符串连接

```
-->a='scilab'; b='--Inria'; c=a+b    //用"+"连接两个字符串
 c    =
```

```
 scilab-Inria
-->s1=string(zeros(2,2));s2=string(ones(2,2));
-->s=s1+s2   //用"+"连接两个字符串数组
 s    =
! 01    01    !
!               !
! 01    01    !
-->s= strcat(['scilab'  ' programming'])  //字符串数组内元素的连接
 s    =
 scilab programming
-->strcat(string(1:10),',')                    //在数组元素之间插入","号
 ans    =
 1,2,3,4,5,6,7,8,9,10
```

3. 求字符串长度

函数调用格式:n=length(M)

该函数可以返回每个字符串的长度。如果参数为字符串矩阵,则结果不再是字符串的长度,而是该字符串矩阵的大小。这与对实数矩阵的操作类似。可以实现此功能的函数还有 size 函数。

例 4.6 求字符串长度和字符串矩阵的大小

```
-->length('scilab')
 ans    =
6.
-->size(['scilab'     ' programming'])
 ans    =
!     1.      2.  !
```

4. 大小写转换

函数调用格式:[y]=convstr(str_matrix, ["flag"])

该函数可以将字符串全部转换为大写或小写。flag 为转换标志,当 flag='u'时,所有字符转换为大写;当 flag='l'时,所有字符

转换为小写。

例 4.7 字符串大小写转换

```
-->A=['this','is';'my','matrix'];
-->convstr(A,'u')
 ans    =
!THIS   IS          !
!                   !
!MY      MATRIX     !
-->convstr(A,'l')
 ans    =
!this    is         !
!                   !
!my      matrix     !
```

5. 字符串的查找

函数调用格式:ind=strindex(str1,str2)

该函数可以查找一个字符串或字符串数组 str2 在另一字符串中的位置 str1,并将 str2 每次在 str1 中出现的首字母位置保存在向量 ind 中。如果 str2 为字符串数组,则分别寻找各元素的出现位置。若不存在则返回空值

例 4.8 查找字符串

```
-->k=strindex('SCI/demos/scicos','/')   // "/"在字符串中出现两
                                        // 次,查询其位置
 k    =
!    4.    10.  !
-->k=strindex('SCI/demos/scicos',['SCI','sci'])  // 分别返回 SCI 与
                                                 // sci 的起始位置
 k    =
!    1.    11.  !
```

```
-->strindex('SCI/demos/scicos','scilab')   //'scilab'在原字符串中不存
                                           // 在,给出空值
 ans    =
      []
```

6. 字符串替换

函数调用格式:str=strsubst(str1,str2,str3)

该函数将原字符串 str1 中的一部分 str2 用另一个字符串 str3 替换掉。如果 str1 不包含 str2,则 str1 保持不变;否则 str2 部分被 str3 替代。

例 4.9 字符串部分替换

```
-->strsubst('the weight is same','weight','height')   // 将其中的
                                                      // "weight"用
                                                      // "height"代替
 ans    =
 the height is same
```

7. 利用字符串实现公式的赋值计算

函数调用格式:A=evstr(str)

该函数将原字符串 str 中的各个已知变量的数值代入公式进行计算。

例 4.10 变量矩阵的赋值计算

```
-->A=['x+y' 'y^w';'z^2+2*z-3' 'w*v']
 A    =
! x+y          y^w    !
!                     !
! z^2+2*z-3    w*v    !
-->x=1;y=2;z=3;w=4;v=5;
-->evstr(A)
 ans    =
!    3.       16.  !
!    12.      20.  !
```

4.1.3 多项式类型

SCILAB 可以方便地处理单变量的多项式计算。这也为完成各种控制问题、信号处理、系统分析等应用提供了方便。下面介绍 SCILAB 中几种常用的多项式操作方法。

1. 多项式表示

通常，可以用如下指令定义多项式的变量'x'：

```
x=poly(0,"x")
```

SCILAB 中的多项式是按升幂顺序显示的。例如多项式 p 可以表示为

$$1 + 2x + x^2$$

这种表示方式与一般的计算机表达式不同，但是 SCILAB 提供了一个转换函数 poly2str。其调用格式为

```
[str]=pol2str(p)
```

该函数将多项式 p 转为字符串 str。

```
-)str=pol2str(p)
 str    =
  1+2 * x+x^2
```

创建多项式的基本指令如下，其调用格式为

```
[p]= poly(a,"x", ["flag"])
```

其中：

a：实数型矩阵或标量；

x：符号变量；

flag:字符串,值为“roots”或“coeff”,缺省值为“roots”。

如果 a 是一个矩阵,那么 p 为特征多项式,即

$$p=\begin{vmatrix} x-a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & x-a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & & \vdots \\ -a_{n1} & -a_{n2} & \cdots & x-a_{nn} \end{vmatrix}$$

如果 v 是个向量,那么 **poly(v,"x",["roots"])**创建根向量为 v 的多项式。**poly(v,"x","coeff")**创建系数为 v 的多项式。缺省情况下是按照多项式的根处理。创建一个多项式可以通过多项式的系数,也可以通过多项式的根实现。

2. 通过多项式的根创建多项式

例 4.11 由根向量生成多项式

以 p=(x-1)(x-2)为例:

```
->a=[1 2];
->p=poly(a,"x")
        p   =
                    2
          2 - 3x + x
```

或者

```
-->x=poly(0,"x")
          x   =
              x
-->p=1+x+2*x^2
          p   =
                    2
      1 + x + 2x
```

3. 通过多项式系数创建多项式

函数调用格式:[p]=inv_coeff(C[,d,[name]])

其中:

C:多项式系数矩阵;

d:生成的多项式的最高次,缺省值为 size(C,'c')/size(C,'r')−1,必须是整数;

name:多项式中变量的名称,例如 x,y 等,缺省为 x;

p:返回的多项式。

例 4.12 通过系数创建多项式

以建立多项式 x^2+2x+1 为例:

```
-->c=[1 2 1];
-->p=inv_coeff(c)
   p    =
                    2
      1 + 2x + x
```

注意其中 x 的幂的表示形式。相反,也可以通过以下指令得到多项式的系数。

函数调用格式:[C]= coeff(p [,v]))

其中:

p:多项式矩阵;

v:正整数,用来指定返回哪些幂项的系数;

C:返回的多项式系数。

例 4.13 提取多项式系数

以 $p=x^2+2x-1$ 为例:

```
-->c=coeff(p)
 c    =
!    1.    2.    1.    !     // 结果返回一个实数向量
```

```
-)c=coeff(p,2)
 c   =
   1.                          //只返回二次项的系数
```

4. 求多项式的根

通过调用 SCILAB 函数可以求多项式的根，该函数名为 roots，其调用格式为

```
[x]=roots(p)
```

其中：

p：需要求根的多项式；

x：多项式的根向量。

例 4.14 求多项式的根

以 $p=x^2-3x+2$ 为例：

```
-)x=poly(0,"x");
-)p=2+3*x+x^2;
-)v=roots(p)
 v   =
! - 1. !
! - 2. !
-)roots(poly(1:15,"x"))
 ans   =

!    1.          !
!    2.          !
!    3.          !
!    4.          !
!    5.          !
!    6.          !
!    7.0000001   !
!    7.9999997   !
```

```
!    9.0000006 !
!    11.000001 !
!    9.9999992 !
!    12.       !
!    13.       !
!    14.       !
!    15.       !
```

由此可以看出各个根值的求解精度。

5. 多项式的乘法和除法

当两个多项式相乘时，乘积的系数向量为这两个多项式系数向量的卷积，而除法为系数向量的解卷运算。多项式之间用“*”就可以实现相乘的符号运算，相当于 MATLAB 中的 conv 函数。多项式除法用 pdiv 函数实现，相当于 MATLAB 中的 deconv 函数。其调用格式为

[R,Q]=pdiv(P1,P2) 或 [Q]=pdiv(P1,P2)

其中：

P1：多项式矩阵；

P2：多项式或多项式矩阵；

R,Q：两个多项式矩阵，分别为商矩阵和余数矩阵。

两个多项式矩阵或者一个多项式矩阵与一个多项式相除，返回商矩阵 Q，或者同时返回余数矩阵 R。矩阵之间满足

$P1_{ij} = Q_{ij} * P2 + Q_{ij}$，当 P2 是一个多项式时

或

$P1_{ij} = Q_{ij} * P2_{ij} + Q_{ij}$，当 P2 是一个多项式矩阵时

例 4.15 多项式的乘法与除法运算

```
-->x=poly(0,'x');                    //指定 x 为多项式符号变量
-->p1=x^2-3*x+2; p2= 2*x^3-x^2-1;
```

```
-->p=p1 * p2                          //多项式 p1 与 p2 相乘,乘积为 p
 p   =
                    2    3    4     5
  - 2 + 3x - 3x + 7x - 7x + 2x
-->[r,q]=pdiv(p2,p1)      //多项式 p2 与 p1 相除,商为 q,余数为 r
 q   =
   5 + 2x
 r   =
  - 11 + 11x
```

试比较以"/"进行的相除运算：

```
-->q2=p2/p1
 q2   =
                2
    1 + x + 2x
  ----------
    -2 + x
```

结果为有理式形式，分子分母之间不可再约。从 SCILAB 的 denom 与 numer 函数可以获得分式的分子与分母部分：

```
-->denom(q2)              //提取分母
 ans   =
  - 2 + x
-->numer(q2)              //提取分子
 ans   =
                2
    1 + x + 2x
```

多项式矩阵之间的运算遵循矩阵之间加减乘除的运算规则，例如用"+"实现矩阵对应多项式元素的相加；用".*"实现对应多项式元素的相乘。

例 4.16 多项式矩阵的运算

```
-->p1=[x-1 x-2;x+1 x+2];
-->p2=[x+1 x-1;x x+2];
-->p1+p2        //多项式矩阵相加
 ans    =
!     2x          - 3 + 2x        !
!                                 !
!     1 + 2x        4 + 2x        !
-->p1. * p2
 ans    =
!           2                 2   !
! - 1 + x       2 - 3x + x      !
!                                 !
!           2                 2   !
!     x + x       4 + 4x + x      !
```

6. 多项式求导

函数调用格式：**pd=derivat(p)**

其中：

p：多项式或有理式矩阵；

pd：对 p 的一阶导数。

例 4.17 多项式求导

```
x=poly(0,'x');
-->p=[x^2+2 * x+1    x^3-1];  //定义一个最高次为 3 的多项式
                              // 矩阵
-->derivat(p)
 ans    =
!                       2 !
!     2 + 2x        3x    !

-->p=1/(x+1)                  //定义 一个有理式
 p    =
```

```
        1
  ---------
    1 + x
-->derivat(p)                    //对有理式求导
 ans    =
          - 1
  ---------------------
                  2
    1 + 2x + x
```

7. 多项式求值

SCILAB 提供 horner 函数计算给定自变量的多项式的值。如果自变量为一个实数，则计算结果为一个数值，在 MATLAB 中功能相似的函数为 polyval；如果自变量是个多项式，则 horner 函数自动进行符号运算，结果仍为一个多项式，在 MATLAB 中功能相似的函数为 subs。

调用格式：[v]=horner(P,x)

其中：

P：多项式或多项式矩阵；

x：自变量的取值，可以是实数或多项式；

v：多项式的值，可以为实数或多项式。

例 4.18 多项式求值

```
-->x=poly(0,'x');                //定义符号变量
-->p=[x-1 x+2];                  //建立多项式向量
-->horner(p,2)                   //令 x 取值为 2
 ans    =
!    1.     4.  !
-->horner(p,%i)                  //令 x 取符号 i
 ans    =
! - 1. + i        2. + i    !
```

```
-)horner(p,x^2)                    //用 x^2 替换 x
 ans    =
!            2            2 !
! - 1 + x      2 + x  !
```

8. 矩阵的特征方程

当 a 是方阵并采用根方式建立多项式时，其输出是该方阵的特征方程。

例 4.19 矩阵的特征方程计算

```
--) poly([1 2;3 4],'s')
 ans    =
              2
  - 2 - 5s + s
```

4.1.4 表(list)类型

前面介绍的各种矩阵内部都是单一类型的数据，但是 list 类型或者说“表类型”的元素可以同时包含各种数据类型，而不是限于单一的某种类型，因此其应用非常灵活。比如，list 类型的元素可以有的为实型数据类型，有的为字符串类型。创建 list 类型数据的指令就是 list，类似于 MATLAB 中的 cell 类型。

1. 创建 list 变量

指令格式：**list(a1,…,an)**

指令说明：a1,…,an 为加入 list 变量的不同数据，可以为任意类型，包括数组和 list 类型。

例 4.20 建立简单的 list 数据

```
--)x=list(12,'string1')    //往 list x 输入一个整数和一个字符串字段
 x    =
       x(1)
```

```
    12.
       x(2)
 string1
-->y=list([0.4 0.8],x)    //list y 的第一个元素为向量,第二个元素为
                           //list x
 y    =
       y(1)
!     .4      .8 !
       y(2)
         y(2)(1)
    12.
         y(2)(2)
 string1
```

这说明 list 类型是可以嵌套使用的。

2. list 数据字段的提取

list 变量的字段可以通过给出字段的索引来调用。

指令格式:[x,y,z,…]=l(v)

指令说明:提取 list 变量 l 中下标值为向量 v 的数据,将结果赋给相应的变量 x,y,z,…。[x,y,z,…]=l(:)将 list 中的所有元素提取出来放在变量 x,y,z,…中。若要进一步提取 list 字段中的字段,则使用多个"()"进行引用。

例 4.21 list 类型数据的提取,接上例

```
-->a=y(1)
 a    =
!     .4      .8 !
-->b=y(2)(2)
 b    =
 string1
```

3. list 数据字段的插入

插入字段，其实就是对原来位置字段的重新赋值。

指令格式：l(i)=a

指令说明：在 list 变量 l 的第 i 位插入数据 a，原数据被替换掉，list 长度保持不变。若是在表头或表尾插入数据，则 list 长度增加。新的数据的类型可以与原数据类型不同。

例 4.22　list 数据元素的插入

```
-->l=list(12,'abcdef');
-->l(1)='test'                    //用"test"替换原来的 12
 l  =
        l(1)
 test
        l(2)
 abcdef
-->[n,s]=l(:)                     //提取全部数据放在矩阵中
 s  =
 abcdef
 n  =
    test
-->l(0)='first'                   //在表头插入数据
 l  =
        l(1)
 first
        l(2)
 test
        l(3)
 abcdef
-->l(4)='last';                   //在表尾增添数据，表长度增 1
-->size(l)
 ans  =
    4.
```

4. list数据字段的删除

删除字段,用"null"指令。

指令格式:**l(i)=null()**

指令说明:清除list变量l的第i位数据,后面的数据向前移动一位,其他数据保持不变,list长度减少一位。求list元素个数可以用**n=size(l)**或**n=length(l)**。

例4.23 list数据元素的删除

```
-)l=list(1,2,3)
 l  =
        l(1)
     1.
        l(2)
     2.
        l(3)
     3.
-)length(l)        //返回list类型元素个数
 ans  =
     3.
-)l(2)=null()   //删除第二个数据
 l  =
        l(1)
     1.
        l(2)
     3.
-)size(l)
 ans  =
          2.
```

5. tlist和mlist类型

tlist也是list类型,但是它的第一个域是个字符向量,定义了

该 list 类型定义的对象名称以及各字段的名称，相当于表头。tlist 类型可以替代 MATLAB 中的 struct 类型。下面通过例子说明这一类型。

例 4.24 tlist 类型

```
-> l = tlist (['guys','name','age'], ['wangfei','zhangming'], [20,
30])  //包括人名、年龄
 l  =
        l(1)
! guys   name   age   !
        l(2)
! wangfei   zhangming   !
        l(3)
!    20.     30.  !
```

该例中 tlist 的第一个元素为字符串向量，说明了该变量存储的对象是“guys”的信息，信息包括“name”和“age”；相应地，第二个元素的内容为“name”，第三个元素的内容为“age”，都是向量。

如果要逐条显示信息，需要通过编写函数实现。关于函数的内容在其他章节有专门叙述，这里只简单给出一个例子：

```
//定义用于改变显示的函数 guys
function guys(l), disp([l.name(:) string(l.age(:))]), endfunction
->guys(l)  //调用该显示方式
! wangfei      20   !
!                   !
! zhangming   30   !
```

结果为逐条显示每人的姓名和年龄。

mlist 类型是 tlist 类型的一种，但是其对字段的引用不是通过整数下标实现，而是通过字段名实现。下面通过例子说明。

例 4.25 mlist 类型

```
-->l=tlist(['guys','name','age'],['wangfei','zhangming'],[20, 30]);
//定义 tlist 类型
-->lm = mlist (['guys','name','age'],['wangfei','zhangming'],[20,
30]); //定义 mlist 类型
-->l(2)   //引用 tlist 类型的"name"字段
 ans   =
! wangfei   zhangming   !
-->lm(2)                  //试图通过整数下标引用 tlist 类型的"name"字
                          //段,出错
        ! --error       4
undefined variable : %l_e
-->lm('name')             //通过字段名"name"引用
 ans   =
! wangfei   zhangming   !
```

从这个例子可以看出 tlist 与 mlist 的区别。

4.1.5 函数

函数文件的扩展名也是".sci",其最重要的一条是文件的第一句可执行语句为 function 引导的定义行。紧跟的通常是"//"引导的注释语句,说明函数的功能、作者等相关信息。然后是函数体。以下为 SCILAB 提供的提取下三角矩阵的函数,文件名为"%sp_tril.sci"。

例 4.26 SCILAB 函数示例

```
function d=%sp_tril(a,k)
// Copyright INRIA
[lhs,rhs]=argn(0)
if rhs==1 then k=0,end
[ij,v,sz]=spget(a)
```

```
m=sz(1);n=sz(2)
l=find(ij(:,1)>=(ij(:,2)-k))
d=sparse(ij(l,:),v(l),[m,n])
```

上例中的函数定义行为

```
function d=%sp_tril(a,k)
```

当函数有多个输入或输出参数时，参数之间用逗号隔开，多个输出参数用方括号[]括起来，例如 **function [stk,txt,top]=sci_max()**。如果没有输入或输出，可以不写相应的参数，例如 **function= %i_plot2d(varargin)**。

SCILAB 本身提供了很多工具箱，这些几乎都给出这样的函数文件，位于“macro”目录下。与 MATLAB 不同的是，SCILAB 定义的函数不能直接使用，在第一次调用前必须先用“getf”命令将其加载到内存中，以后可以重复调用。

例 4.27 函数的定义与使用

(1) 定义一个简单的函数，输入一个整数，返回该数的平方。

```
function a=test(b)
//test on SCILAB function
a=b*b;
```

将这些内容保存在文件“exe_fun”中。

(2) 加载以及调用函数

```
-->getf('C:\sci\exe_fun.sci');    //加载该函数于内存中
-->a=test(3)                      //调用该函数
 a=
    9.
```

因此在用 SCILAB 编程时通常编写一个“loader.sci”，用来加载所有要用到的自定义函数。

另外一种定义函数的方法是用关键字 endfunction 在线定义，

在这种情况下不需要用“getf”加载，可直接使用。用 function/endfunction 可以在程序的任意位置定义函数，使用起来非常方便。

```
-)function a=two(), a=2, endfunction
-)b=two()+1
 b=
      3.
```

要注意，如果在调用的时候不加()，则结果只是复制一个函数。

```
-)x=two          //复制函数 two,生成功能相同的函数 x
 x    =
[a]=x()
-)b=x()+2        //调用新函数 x
 b=
      4.
```

另外 SCILAB 也支持函数的递归调用，当然递归的深度有限制。

4.2 程序控制语句

除了顺序执行语句以外，与其他编程语言一样，SCILAB 提供了循环语句、条件判断语句等控制程序流程的语句，应用起来比较灵活。下面介绍这类语句的用法。

4.2.1 循环语句

与 MATLAB 一样，SCILAB 可以通过 for 语句和 while 语句实现循环，而且格式也很相似。不同的一点是，当 for 语句与循环体写在同一行时，在循环条件后面必须有关键词“do”或者逗号“,”。

1. for 语句

语句格式：

for 循环变量=初值:步长:终值

循环体

end

循环变量步长的缺省值为 1，也可以自己指定或正或负的实数。每一个 for 与一个 end 对应。下面是使用 for 语句的几个例子。

例 4.28 for 语句的使用

```
for i = 1 : 10
   f(1,i)=i
end
-->f
 f  =
!    1.   2.   3.   4.   5.   6.   7.   8.   9.   10.  !
--> for i = 10 : 1, g(1,i)=i;end   //for 语句写在一行，用“,”隔开
-->g
  ! --error       4
undefined variable : g                              //循环体未被执行
--> for i = 10 : -1 : 1, g(1,i)=10-i+1;end   //循环变量递减
-->g
 g  =
!    10.   9.   8.   7.   6.   5.   4.   3.   2.   1.  !
```

for 语句还可以嵌套使用，如下例：

```
for i = 1 : 2
   for j = 1 : 5
      m(i,j) = i+j
   end
end      //for 语句嵌套
```

```
-->m
 m    =
!    2.    3.    4.    5.    6.  !
!    3.    4.    5.    6.    7.  !
```

另外，矩阵的列数与 list 变量的元素个数也可做循环次数控制。

例 4.29 以 list 变量的元素个数控制循环次数。

```
l=list('abc','def');        //定义一个 list 类型变量 l
for  e=l                    //循环变量依次取 l 的各个元素值
       e
end
 e    =
 abc
 e    =
 def
```

对矩阵也类似。

2. while 语句

语句格式：

```
while  表达式
    循环体
end
```

首先判断表达式，如果表达式值为真，则执行循环体部分一次；再来判断表达式是否为真。这样重复进行，一直到表达式值为假，就跳过循环体部分，向下继续执行。如果表达式开始时为真，为避免死循环，循环体内应有改变表达式值的语句。While 循环可以用于循环次数事先未知的情况。

例 4.30 求平方值小于 1000 的最大整数

```
j=1;
```

```
while j * j <1000
   j=j+1;
end
->j
 j   =
       32.      // 平方值小于 1000 的最大整数应该是 31(j-1)
```

4.2.2 判断语句

1. if 语句

当关键词 if 与指令分行写时，指令的基本格式如下：

```
if  表达式
     语句序列;
end
```

该语句的功能为如果表达式结果为真，则执行后面的语句序列；如果表达式结果为假时还需要别的操作，则书写格式如下：

```
if  表达式
     语句序列 1;
else
    语句序列 2;
end
```

当表达式结果为假时执行 else 后面的语句。这种书写格式与 MATLAB 一样。但是当指令与 if 语句写在同一行时，表达式后面必须有 then 或逗号“,”，例如：

```
if  表达式 then      语句序列; end
if  表达式 ,         语句序列;  end
```

If 语句的嵌套形式为

```
if  表达式 1 then  语句序列 1
 elseif  表达式 2 then  语句序列 2
```

```
    ⋮
 else   语句 n
end
```

注意每个 if 必须有一个 end 与之配对，否则出错。

例 4.31 判断一个 0～1 之间的随机数是否大于 0.5，若是，输出 1；否则输出 0。

```
s=rand()
if s>0.5
   disp('1')
else
   disp('0')
end
```

结果为随 s 值的每次取值不同而变化。

判断语句与循环语句结合在一起使用可以灵活地解决问题。

例 4.32 输出平方值小于 1000 而且大于 10 的整数。

```
j=1;
while j*j<1000
  j=j+1;
  if j>10
     disp(j);
  end
end
```

2. select-case 语句

select-case 语句用于有多种备选的情况，类似于 MATLAB 中的 switch-case 语句。与前面 if 语句相同，当指令与 case 语句写在同一行时，必须加上 then 或“,”，其语句格式如下：

```
select  表达式 0,
  case  表达式 1 then  指令 1,
```

```
    case 表达式 2 then 指令 2,
         ⋮
    case 表达式 n then 指令 n,
    [else 其他指令],
end
```

语句说明：判断哪个表达式的值与表达式 0 相等，然后执行相应指令。

例 4.33 产生一个 0～3 之间的随机数，输出相应结果。

```
n=round(3 * rand());
select n
case 0 then disp('0')
case 1 then disp('1')
case 2 then disp('2')
else disp('3')
end
```

在 MATLAB 中，switch 表达式的类型只能是标量或字符串，但在 SCILAB 中，select 表达式可以是任意类型，关键词 otherwise 用关键词 else 替代。

4.3 SCILAB 脚本文件应用

4.3.1 脚本文件的建立与调试

如第 2 章所述，SCILAB 有两种常用方式：一种是直接交互的指令行操作方式；另一种是程序文件方式。SCILAB 程序文件为 ASCII 编码的脚本文件，可以用任意字处理软件建立、编辑和修改。它是扩展名为“.sci”，的解释型执行方式语言。我们建议用户从网上下载自由软件 PFE 来完成脚本文件工作（参见附录 A

中的网址)。

脚本是一些函数和命令的组合,没有输入输出参数,可以在SCILAB环境中直接执行。生成的变量为全局变量,保存在内存中,直到用clear命令将其删除。前面给出的例子中的程序段都是脚本。虽然SCILAB提供交互式的命令窗口,当用户要发出的指令太多时,就比较适合建立一个脚本文件,将所有命令先保存起来,以便一次执行,就像DOS下的批处理命令一样。如下的例子介绍了建立一个SCI文件的过程。

例4.34 建立和执行SCI文件

(1) 通过字处理软件建立一个简单的脚本文件,要求输出SIN和COS曲线。这里使用的编辑软件是PFE。把文件保存为"examples.sci"(见图4.1)。

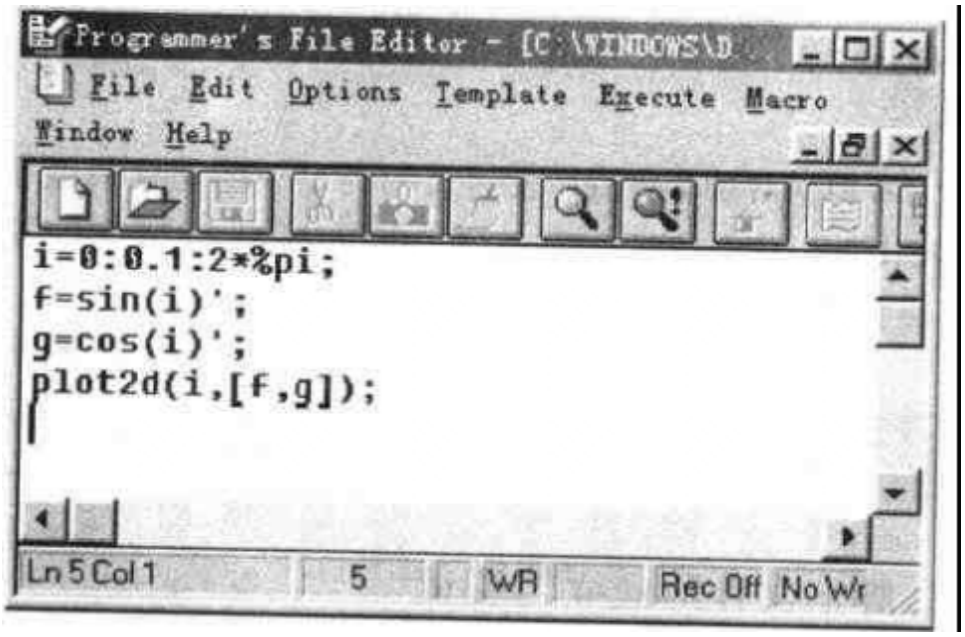
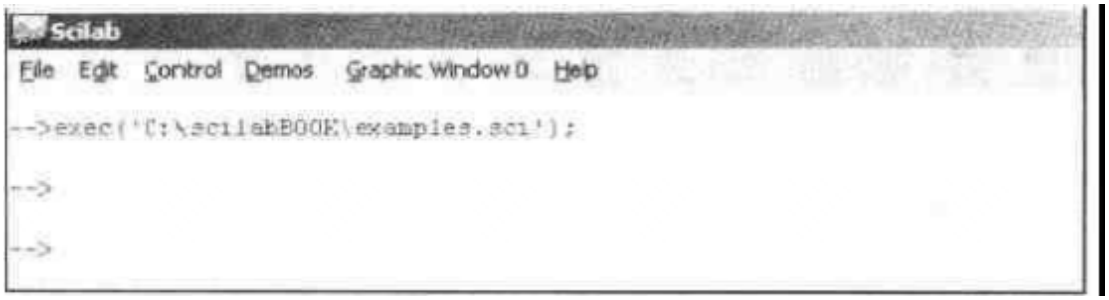


图4.1 利用PFE编辑SCILAB脚本文件

(2) 在SCILAB中执行"examples.sci"文件(见图4.2)。

执行完后将开出一个图形窗口显示两条曲线,而且向量i, f, g还在内存中,可以在指令窗口输入变量名,以查看其值。


Scilab
File Edit Control Demos Graphic Window 0 Help
-->exec('C:\scilabBOOK\examples.sci');
-->
-->
-->exec('C:\scilabBOOK\examples.sci');
-->
-->
-->exec('C:\scilabBOOK\examples.sci');
-->
-->

```
-1->a
 a    =
    2.
-1->return
-->a
 a    =
    1.
```

pause指令在编程调试的时候非常有用。在程序运行过程中，可以直接通过键盘的“Ctrl-c”实现暂停，以中止那些无限循环。“abort”指令中止暂停状态，退出调试。

4.3.2 函数参数与工作空间

函数调用的过程包含了参数传递的过程。比如在调用SCILAB的求下三角矩阵的函数tril时，

```
-->a=[1 2 ;3 4];
-->b=tril(a)
```

a为输入的实际参数；b为输出参数。参数传递是函数与SCILAB工作空间数据传送的通道。

在SCILAB函数调用时，通过固定变量argn可以实现任意数目参数的输入与输出，其功能类似于MATLAB中的nargin和nargout，目的是实现函数参数传递的可调性。函数通过判断输入参数的个数做相应的处理。例如上面的tril例子就是对输入参数的个数作了判断。要说明的是，虽然输入的参数的个数可以变，但是输入的顺序不可以变。

不同的函数有不同的工作空间，函数内所创建的变量只存储在函数内部的工作空间，调用结束以后就消失，是局部变量。变量的工作环境是嵌套的，最里层为最后调用的函数。与MATLAB不同的是，在里层函数内可以读取外层函数的变量，但不能修改，

也就是不能把变化的结果返回。当函数内有与 SCILAB 工作空间同名的变量时，函数内部定义的变量起作用。比如在函数中定义

```
function test()
a="this is a string in function"
disp(a)
```

加载该函数后，在 SCILAB 中建立另一变量 a，并调用该函数，可以看出同名变量 a 具有两个不同的值。

```
-->a=2;
-->test()                    //调用函数 test
 this is a in function       //显示在函数工作空间的变量 a 的内容
-->a                         //在 SCILAB 工作空间的变量 a 值保持不变
 a    =
       2.
```

SCILAB 有两类工作空间，标准的工作空间为前面所说明的那样，另外一种为全局的工作空间。当希望在不同工作空间中，或不同函数调用中都共享同一变量时，可以应用指令“global”来声明该变量。这点与 MATLAB 非常相似。下面通过一个简单的例子来说明。

例 4.35 应用“global”指令实例

(1) 首先定义函数，其中声明了全局变量“a”。

```
function test()
global a;
a="this is a string in function"
disp(a)
```

(2) 加载函数，并在调用该函数之前声明同名变量“a”。

```
-->global a
```

```
-->a=2;
-->test()
 this is a string in function
-->a
 a    =
 this is a string in function
```

从结果可以看到，作为全局变量的“a”在函数调用结束后改变了值。因此全局变量指令也可以看作函数之间传值的通道，只是由于变量的值可以在很多地方改变，因此使用起来要比较谨慎。

如第2章所述，要获得工作空间的信息，SCILAB 提供了两个函数。函数“who”可以显示标准空间与全局空间内的变量名以及内存的使用情况。若要单独显示局部变量或全局变量及所占用的内存空间，可以使用 who('local')或 who('global')。但是如果要显示每个变量的详细信息，可以使用“whos”命令。“whos”缺省的命令是显示所有变量的详细信息，但是也可以根据类型来显示。

```
-->whos -type boolean
```

Name	Type	Size	Bytes
MSDOS	boolean	1 by 1	24
%F	boolean	1 by 1	24
%T	boolean	1 by 1	24
%t	boolean	1 by 1	24
%f	boolean	1 by 1	24

标准工作区是预先分配了大小的数组，如果在运算中工作空间的大小不够，则显示错误信息。这种情况下用户可以用“stacksize”指令分配更多的内存后重新计算。“stacksize”函数的参数是一个64位的数字。

```
-->a=ones(1000,1000);                    //定义的数组太大引起错误
```

```
                    ! --error     17
stack size exceeded! (Use stacksize function to increase it)
Memory used for variables :      6772
Intermediate memory needed:   1000002
Total   memory available   :   1000001
-->stacksize(2.5 * 10^6)                    //分配内存
-->a=ones(1000,1000);
```

小结：本章介绍了 SCILAB 中的几种特殊类型的矩阵，包括布尔矩阵。字符串矩阵、多项式矩阵及相关运算、List 类型数据及相应操作等。并介绍了与编程有关的内容——函数定义、循环和判断语句等。

第5章

计算结果可视化

5.1 SCILAB 图形窗口介绍

俗话说:“一图胜千言”。人们很难直接从一大堆原始的离散数据中理解它们的含义,而数据图形却能使人快捷、直观地理解数据的许多内在本质。SCILAB 具有良好的数据可视化功能,可以将数据用二维或三维图形表现出来。

在 SCILAB 中能用多个图形窗口显示图形,SCILAB 自动将其命名为 ScilabGraphicx,其中 x 是窗口的编号。在任何时刻只能有一个窗口被激活。在 SCILAB 的运行主窗口中,用菜单中的“Graphic Window x”项来管理图形窗口(x 表示当前激活窗口的编号)。利用该菜单能新建、弹起或删除第 x 窗口。用户可以直接建立任意标号的窗口。如果必要,则绘图指令会自动建立一个新的图形窗口。

在 SCILAB 图形窗口中(见图 5.1),其菜单共包含 4 个按钮:

【File】　　【文件】

Clear　　用于清除图形窗口中已有的图形,以免影响图形窗口下一次的图形绘制。

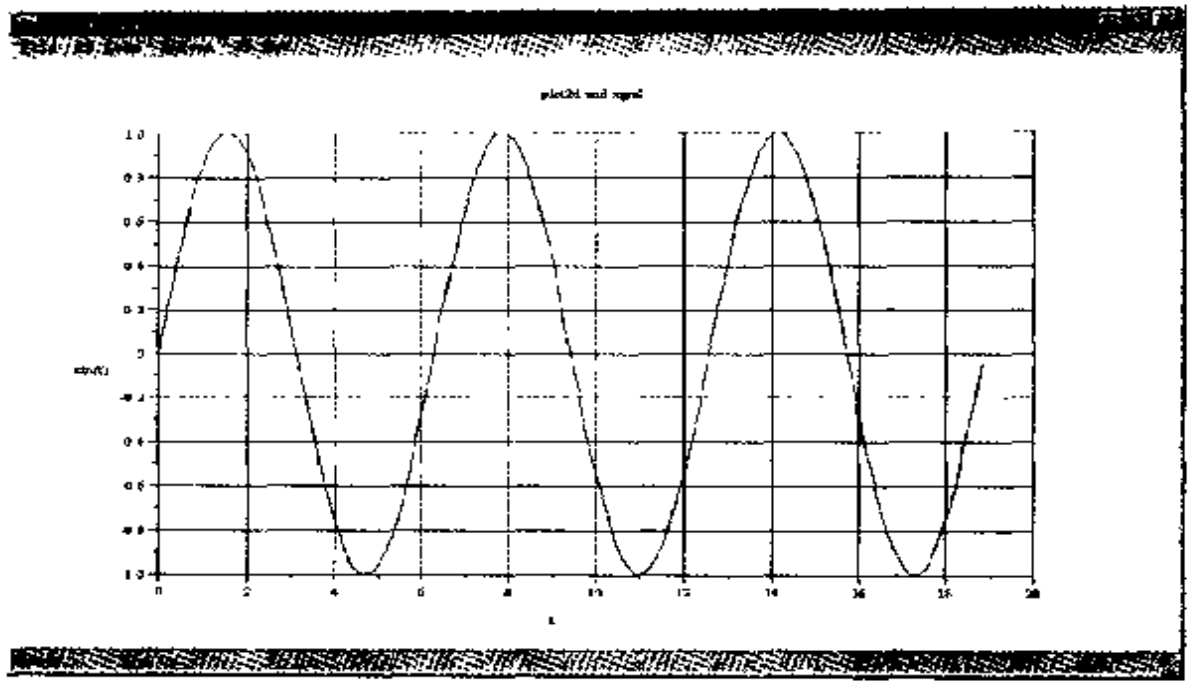


图 5.1 SCILAB 图形窗口示意图

Print... 打开一个打印设置面板。在 UNIX 下，打印机在主 SCILAB 脚本 SCIDIR/bin/SCILAB 中定义，通过 "make all" 从初始的文件 SCIDIR/bin/SCILAB.g 中得到。

Export 打开一个选择面板，将所绘的图形用指定的格式 (Postscript, Postscript-Latex, Xfig) 保存到一个文件中。

Save 直接将绘图保存到一个指定名字的文件中。这个文件能随后载入 SCILAB 中，重新绘图。

Close 与主窗口菜单的 Delete Graphic Window 命令相同，删除图形窗口。

【2D Zoom】 【二维图形的缩放】

这个命令能重复调用。对于三维绘图，这个按钮虽然不被禁止，但不起作用。

【UnZoom】 【返回初始的绘图状态】

这个命令对应上一次缩放，以免多次缩放。

【3D Rot】 【三维旋转】

拖动鼠标旋转所绘制的三维(3D)图形。这个按钮不能用于二维绘图。为了帮助操作(旋转特定的角度)，旋转的角度在窗口顶部显示。

5.2 二维图形的绘制

5.2.1 plot 指令

plot 是最基本的二维图形绘制指令，它属于 SCILAB 的内部函数。本节将比较全面地介绍这个指令的用法。下面介绍 plot 指令绘制二维图形时的基本格式：

(1) **plot(Y)**

若 Y 为向量，则以 Y 元素值为纵坐标，以相应元素下标为横坐标值绘制连线图；若 Y 为实数矩阵，则按行绘制每行元素值并对应列下标的连线图，图中的曲线数等于 Y 阵的行数。

(2) **plot(X,Y)**

若 X 和 Y 是两个向量，则其中 Y 是 X 的函数；在 X 没有给定的情况下，用向量(1,size(y,'*'))代替横坐标值；若 Y 是一个实数矩阵，则按行绘制每行元素的连线图；若 X,Y 是同维矩阵，则以 X,Y 对应行元素为横纵坐标分别绘制曲线，曲线条数等于矩阵的列数。

(3) **plot(X,Y,[xcap,ycap,caption])**

这里 X 和 Y 意义同上。xcap,ycap 和 caption 分别代表图形的 x 坐标标记、y 坐标标记和图形标题，都属于字符串类型。

例 5.1 单向量输入格式，画连线图(见图 5.2)

```
Y = [2,4,5,7,8,0,9,4];
plot(Y)
```

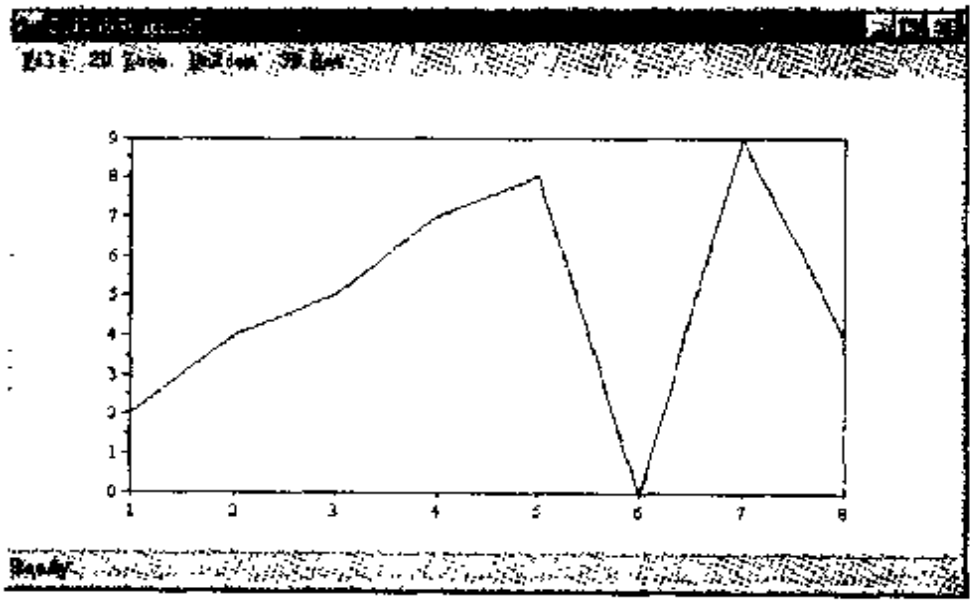


图 5.2 单向量输入格式所画的连线图

例 5.2 双向量输入格式，画曲线(见图 5.3)

```
X=0 : 0.1 : 2 * %pi;
plot(X,sin(X));
```

说明：本例第一条赋值语句使 X 成为行向量，这是最常见的绘制图形生成方式。可以用 plot(sin(X)) 指令代替 plot(X, sin(X))指令，效果完全一样。

例 5.3 在输入量一个是向量，一个是矩阵的情况下，以相同横坐标画多根曲线(见图 5.4)

```
X=0 : 0.1 : 2 * %pi;
plot(X,[sin(X);cos(Y)])
```

说明：[sin(X); cos(Y)]组成一个矩阵，plot(X,[sin(X);

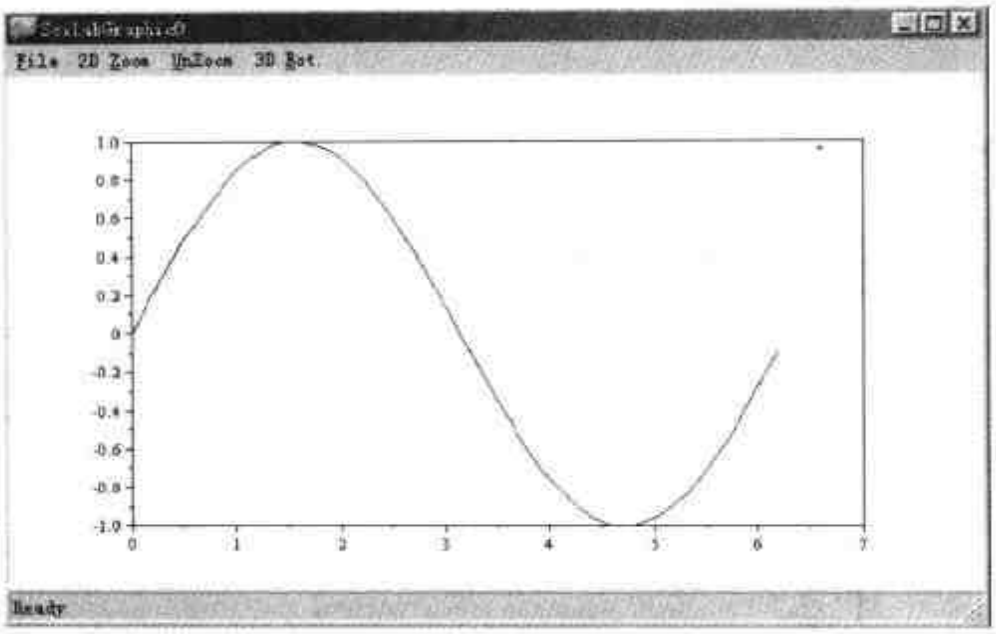


图 5.3 双向量输入格式所画的曲线

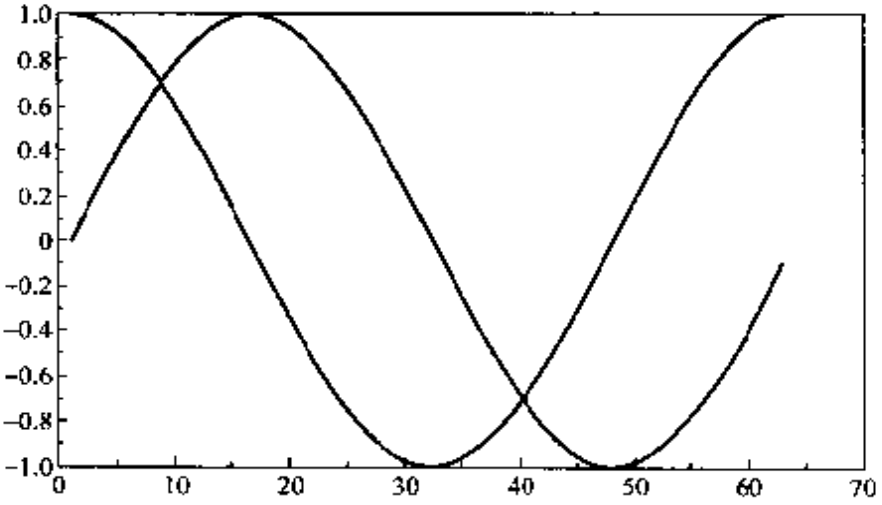


图 5.4 以相同横坐标绘制多条曲线

cos(Y)]也可以用 plot([sin(X);cos(Y)])代替,效果一样。

例 5.4 加注坐标标记和图形标题(见图 5.5)

```
x=0:0.1:2*%pi;
plot(x,sin(x),"sin","time","plot of sinus")
```

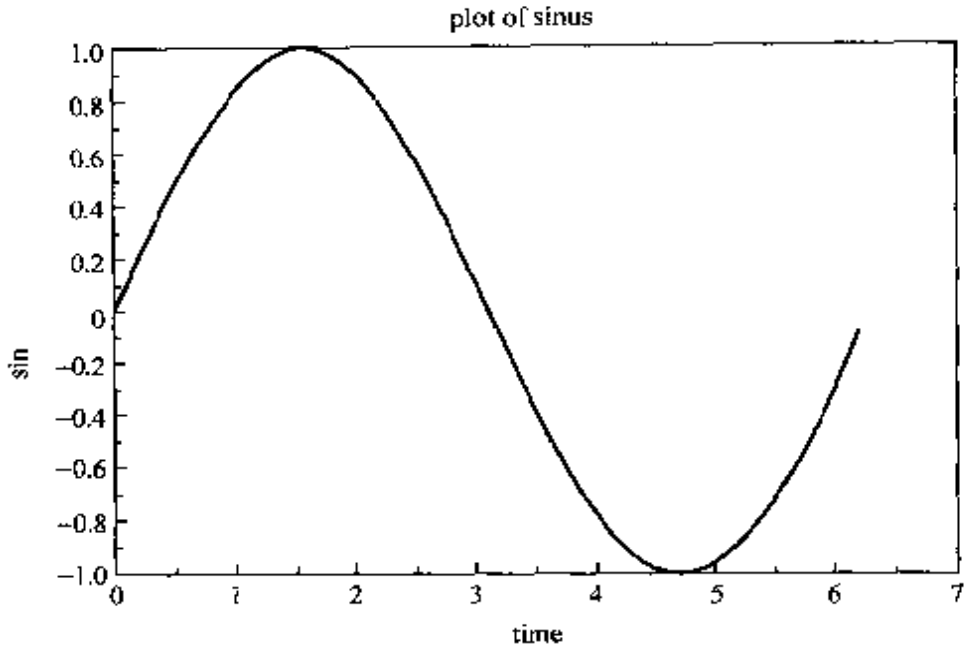


图 5.5 加注坐标标记和图形标题

5.2.2 plot2di 指令

较常用的二维绘图指令是 plot2di，它包含一组绘图指令：

plot2d：用分段连线绘图；

plot2d2：用分段常量方式绘制图形；

plot2d3：绘制柱形图；

plot2d4：用箭头绘图。

绘制二维图形时，具体的 plot2di 指令的基本格式如下：

(1) **plot2di([x,]y)**

x 和 y 可以是两个数值矩阵或向量。如果 y 是一个向量，x 必须是一个具有相同元素个数的向量。在 x 没有给定的情况下，用向量(1,size(y,'*'))代替 x；如果 y 是一个矩阵，x 可以是一个与 y 行数相同的向量，或与 y 具有相同尺寸的矩阵，y 的每列根据相关的 x 绘制。

(2) **plot2di([x,]y,<opt_args>)**

x 和 y 的意义同(1),<opt_args>表示参数声明序列 key1 = value1,key2 = value2,…,这里 key1,key2,… 可能是下列参数之一:

style:该行向量给每个曲线设置线型。第 i 条曲线的线型由 style(i)定义。如果style(i)小于等于 0,对应曲线是用第 id=abs(style(i))的标识方式绘制。标识可用 xset()设置,具体见 xset()的说明。如果 style(i)是正数,曲线用具有颜色的直线或圆点线绘制,颜色的 id= style(i)。

leg:设置曲线的标题。

rect:设置绘图范围。

nax:设置网线。

Logflag:设置坐标轴的刻度类型(线性或对数型)。

Frameflag:指定计算绘图框架的方式。这个参数是一个范围在 0～8 的整数。

Axesflags:指定绘图周围的轴的类型。这个参数是一个取值范围为 0～5 的整数。

例 5.5 利用 plot2d 指令绘制二维图形

```
x=[0:0.1:2*%pi]';
plot2d(sin(x))
```

说明:本例所生成的图形同图 5.3,**plot2d(sin(x))**指令可以用 **plot2d(x,sin(x))**代替。

例 5.6 同时绘制多条曲线(见图 5.6)

```
x=[0:0.1:2*%pi]';
plot2d(x,[sin(x) sin(2*x) sin(3*x)])
```

例 5.7 应用多种图形参数绘图(见图 5.7)

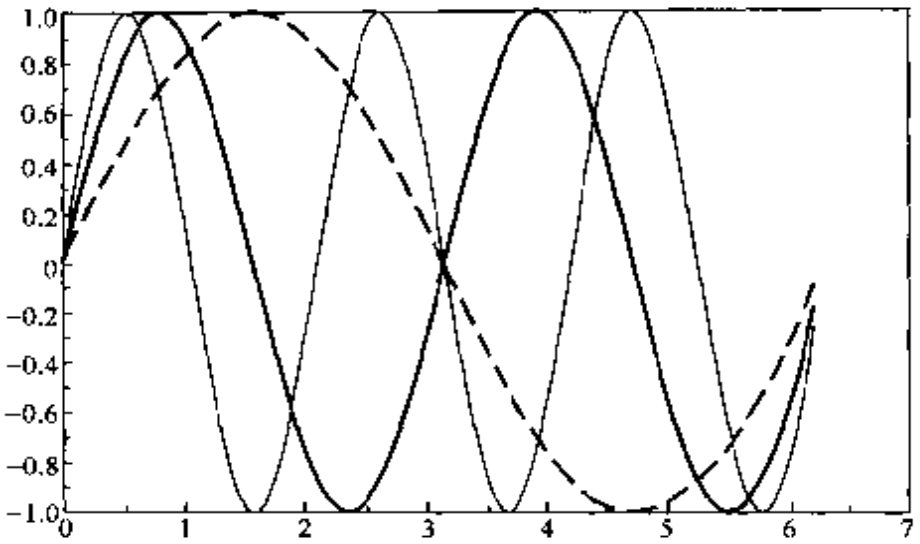


图 5.6 同时绘制的多条曲线(为区别不同的曲线,曲线颜色 SCILAB 自动给定,本书无法显示)

```
plot2d(x,[sin(x) sin(2 * x) sin(3 * x)],…
[1,2,3],leg="L1@L2@L3",nax=[2,10,2,10],rect=[0,-2,2 * %
pi,2])
```

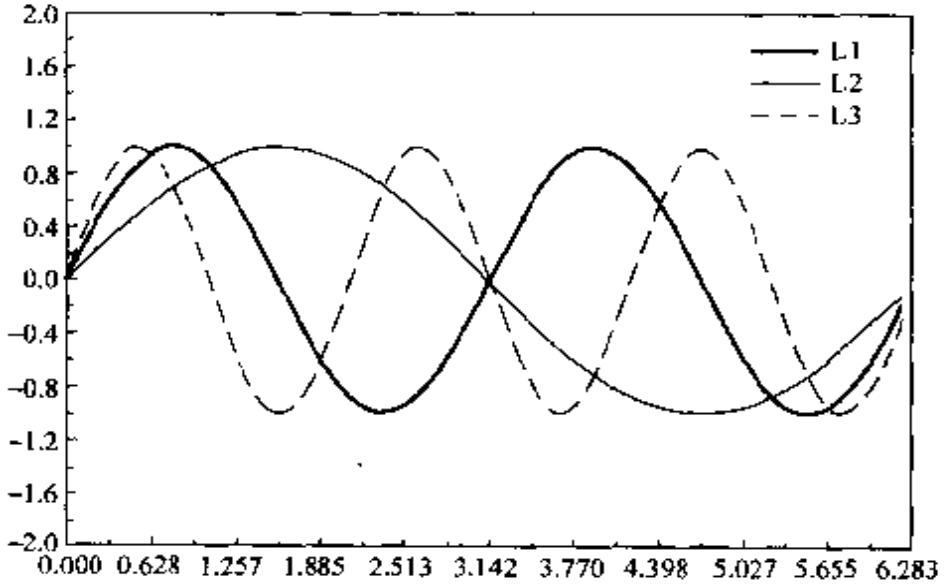


图 5.7 应用多种图形参数的绘图

例 5.8 用分段常量方式 plot2d2 函数绘图示例(见图 5.8)

```
x=[0 : 0.1 : 2 * %pi]';
plot2d2(x,[sin(x) sin(2 * x) sin(3 * x)])
```

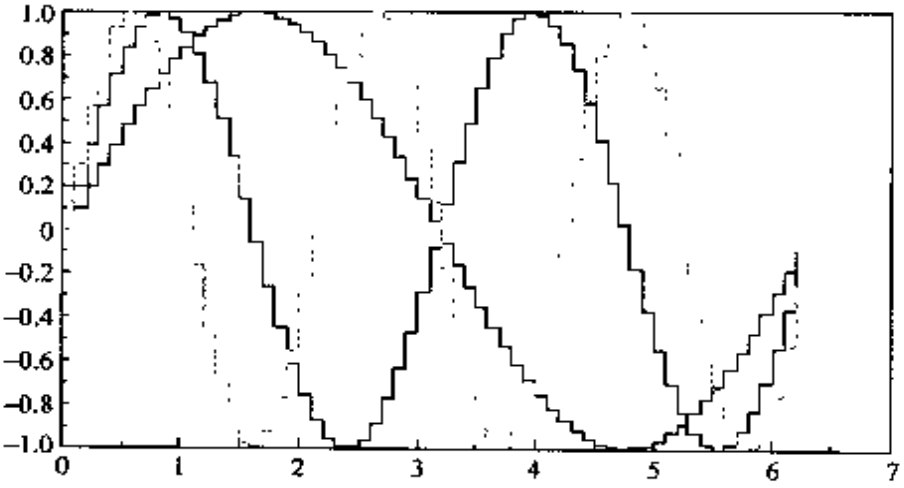


图 5.8 plot2d2 函数(分段常量方式)绘图示例

例 5.9 用柱形图方式 plot2d3 函数绘图示例(见图 5.9)

```
x=[0 : 0.1 : 2 * %pi]';
plot2d3(x,[sin(x) sin(2 * x) sin(3 * x)])
```

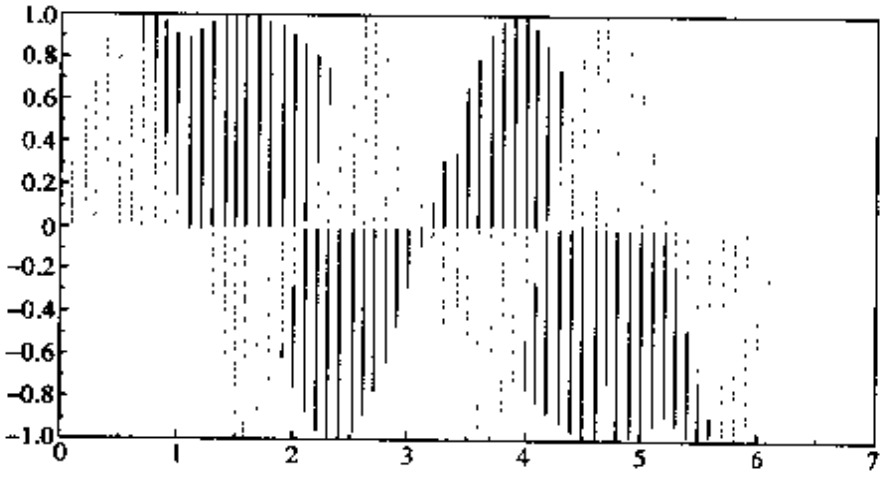


图 5.9 plot2d3 函数(柱形图方式)绘图示例

例 5.10 plot2d4 函数绘图示例(见图 5.10)

```
x=[0 : 0.1 : 2 * %pi]';
plot2d4(x,[sin(x) sin(2 * x) sin(3 * x)])
```

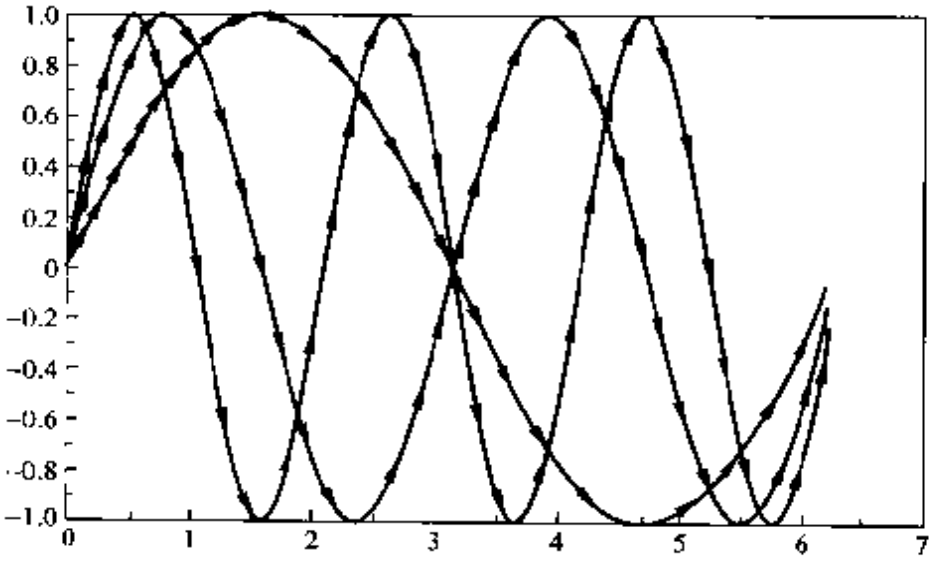


图 5.10 plot2d4 函数(箭头绘图方式)绘图示例

5.3 三维图形的绘制

5.3.1 三维曲面的绘制

SCILAB 在三维空间中绘制三维曲面的基本指令是 plot3d。像 plot 指令一样,它也是一个内部函数。在 SCILAB 中,输入指令 plot3d()可以观看该函数的演示。下面介绍用 plot3d 指令绘制三维图形时的基本格式。

(1) **plot3d(x,y,z[,theta,alpha,leg,flag,ebox])**

用该指令绘制 z=f(x,y)的曲面。

其中:

参数 x,y 表示 x 轴和 y 轴的行向量,必须是单调递增的。如果 x 是长度为 n1 的行向量,y 是长度为 n2 的行向量,则 z 是 n1×n2 的矩阵,z(i,j)是点(x(i),y(j))上的曲面值。

参数 theta 和 alpha 是观察点(视点)的球坐标,是以角度为单位的实数值。

参数 leg 定义每个坐标轴的标题,@是轴标题的分隔符。例如 X@Y@Z,表示 3 个坐标轴的标题分别是 X,Y 和 Z。

参数 flag 是包含 3 个参数的向量,flag=[mode,type,box],其中每个参数的具体意义如下:

mode:图形隐含部分的处理方法。

- mode>0 去除曲面的隐含部分,曲面用颜色模式绘制,mode 是所用颜色的索引号,具体见 getcolor 函数;
- mode=0 绘制曲面的隐含部分;
- mode<0 只用颜色或模式 id 绘制曲面的阴影,每个 id 的意义参见 xset()函数。

type:图形缩放比例。

- type=0 利用当前缩放比例绘图(由先前调用 param3d,plot3d,contour 或 plot3d1 等函数时设定的);
- type=1 用 extreme aspect ratios 自动调节 3d 包络盒的比,边界由 ebox 的值指定;
- type=2 用 extreme aspect ratios 自动调节 3d 包络盒的比,边界用给定的数据计算;
- type=3 3d 包络盒由 ebox 参数设定,与 type=1 类似;
- type=4 3d包络盒由给定的数据设定,与type=2类似;
- type=5 3d 扩展包络盒由 ebox 参数设定,与 type=1

类似；

type=6 3d扩展包络盒由给定的数据设定，与type=2类似。

box：三维图形周围的包络盒参数。

box=0 围绕图形的包络盒不画；

box=1 与box=0同；

box=2 仅画曲面后面的轴；

box=3 绘制包围曲面的包络盒，添加每个轴的标题；

box=4 绘制包围曲面的包络盒，添加轴和标题。

参数ebox：当Flag中的type标记为1时使用。用向量[xmin,xmax,ymin,ymax,zmin,zmax]指定绘制图形的边界。

(2) **plot3d(x,y,z,<opt_args>)**

这种调用格式与第一种类似，参数<opt_args>表示一系列的声明key1=value1,key2=value2,… 这里key1,key2,…可能是上面的参数theta, alpha, leg,flag,ebox中之一。可选择的参数theta,alpha,leg,flag,ebox用一个声明序列key1=value1,key2=value2的方式来定义，这样就不需要特别遵守在上一种调用方式中规定的各个参数的顺序了。

(3) **plot3d(xf,yf,zf[,theta,alpha,leg,flag,ebox])**

用这种调用方式绘制由一组面片组成的曲面。通过替换由[xf1 xf2 …],[yf1 yf2…]和[zf1 zf2…]组成的矩阵，能绘制多个图。参数xf,yf和zf分别表示大小为nf×n的矩阵，用于定义组成曲面的小面片。共包含n个小面片，每个面片i用一个具有nf个点的多边形定义。第i个面片上的x,y,z轴上的坐标由xf(:,i), yf(:,i) and zf(:,i)分别给定。其余参数与上面相同。

(4) **plot3d(xf,yf,zf,<opt_args>)**

这种调用方式的参数分别与上面调用方式中的参数

意义相同。

(5) **plot3d(xf, yf, list(zf, colors)[, theta, alpha, leg, flag, ebox])**

利用 list(zf,color),而不是 zf,能给每个面片设定一特定的颜色。这里 colors 是 n 维向量。如果 colors(i)是正的,则它给出面片 i 的颜色,面片边缘用当前的线型和颜色绘制;如果 color(i)是负的,则使用颜色索引绝对值。应用 xset()方式可以找到对应各种颜色的索引值。

面片的颜色也可以通过内插得到。在这种情况下,colors 必须是一个大小为 nf×n 的矩阵,给定每个面片边界的颜色。面片的颜色由边缘颜色内插得到。

(6) **plot3d(xf,yf,list(zf,colors),<opt_args>)**

这种调用方式的参数分别与上面调用方式中的参数意义相同。

例 5.11 绘制 z=f(x,y)三维曲面(见图 5.11)

```
t=[0:0.3:2*%pi]';
z=sin(t)*cos(t');
plot3d(t,t,z)
```

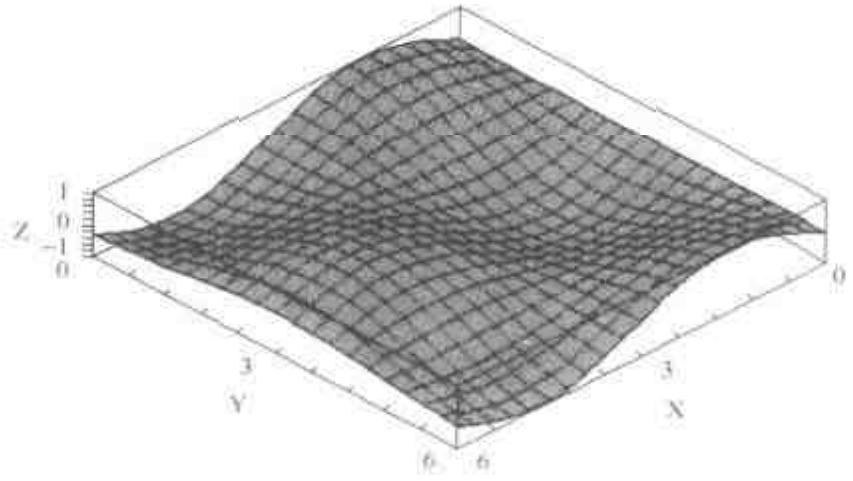


图 5.11 绘制 z=f(x,y)三维曲面

例 5.12 利用函数 genfac3d 产生的图形面片绘制相同的图形(见图 5.11)

```
[xx,yy,zz]=genfac3d(t,t,z);
plot3d(xx,yy,zz)
```

说明:可用函数 genfac3d 计算出组成三维曲面 z=f(x,y)的每个具有 4 个边的面片的坐标。本例中,xx,yy 和 zz 分别是 4×400 的矩阵,分别表示面片的 x,y 和 z 坐标。这 3 个矩阵对应的每一列生成一个具有 4 个顶点的面片,总共有 400 个面片。

例 5.13 同时绘制多个三维图形,并且用不同的颜色(见图 5.12)

```
plot3d([xx xx],[yy yy],list([zz zz+4],…,[4 * ones(1,400),5 * ones
(1,400)]))
```

说明:参数 xx,yy 和 zz 的取值同上。[4 * ones(1,400),5 * ones(1,400)]表示将组成两个曲面的 400 个面片的颜色分别设定为颜色 4 和颜色 5。

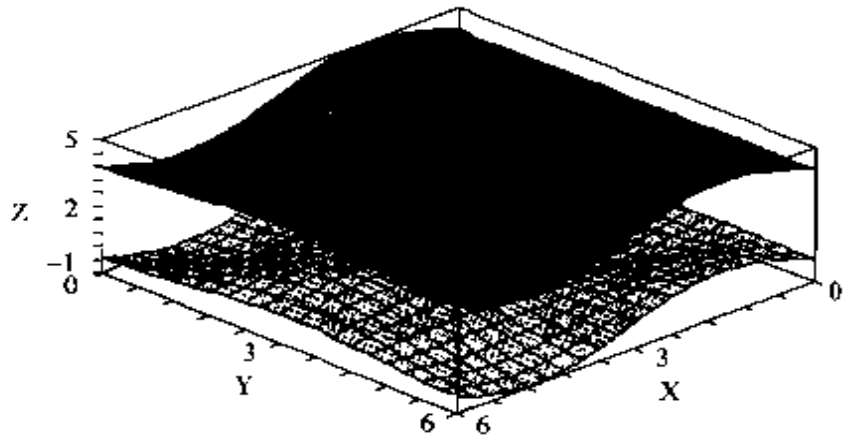


图 5.12 同时绘制多个三维图形,并且用不同的颜色

例 5.14 绘图时使用视点以及添加坐标轴的标题(见图 5.13)

```
plot3d(1:10,1:20,10*rand(10,20),35,45,"X@Y@Z")
```

说明：本例中视点的位置是(35,45)，x，y 和 z 坐标轴的标题分别设为 X，Y 和 Z。

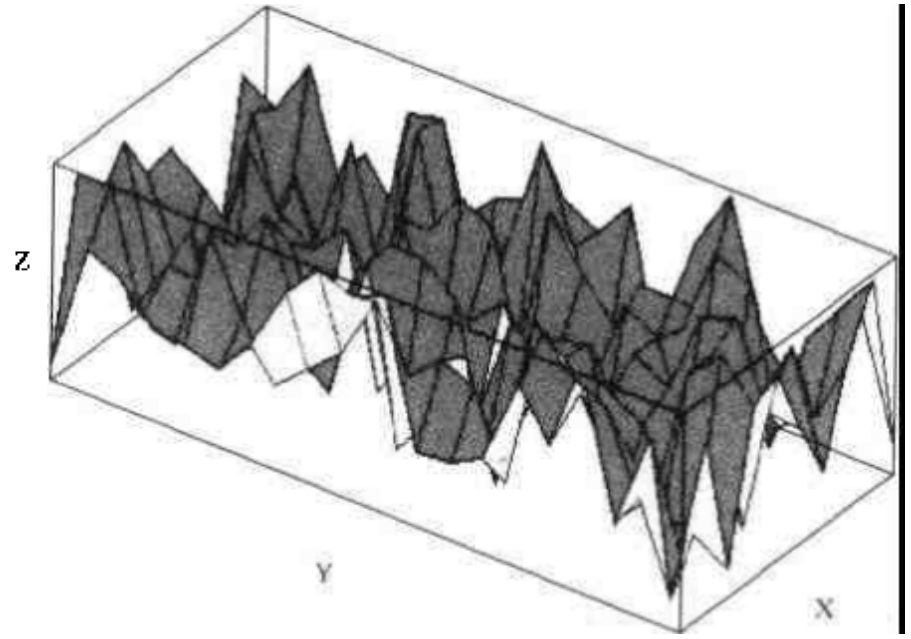


图 5.13 使用视点以及添加坐标轴标题的图形

5.3.2 三维曲线的绘制

1. param3d 函数

函数 param3d 用于绘制由坐标向量 x，y，z 定义的空间参数曲线。输入指令 param3d() 可以观看该函数的演示。下面介绍用 param3d 函数绘制三维图形时的基本调用格式如下：

```
param3d(x,y,z,[theta,alpha,leg,flag,ebox])
```

其中：

参数 x，y 和 z 是具有相同大小的向量，分别表示确定参数曲线各点的 XYZ 坐标。

参数 theta 和 alpha 是观察点(视点)的球坐标，以角度为单位

的实数值。

参数 leg 定义每个坐标轴的标题，@是轴标题的分隔符。例如 X@Y@Z，表示 3 个坐标轴的标题分别是 X，Y 和 Z。

参数 flag 是包含 2 个参数的向量，flag＝[type，box]，其中每个参数的具体意义如下：

① type：整数变量，用于设定图形缩放比例。

type＝0　利用当前缩放比例绘图（由先前调用 param3d，plot3d，contour 或 plot3d1 等函数时设定的）。

type＝1　用 extreme aspect ratios 自动调节 3d 包络盒的比，边界由 ebox 的值指定。

type＝2　用 extreme aspect ratios 自动调节 3d 包络盒的比，边界用给定的数据计算。

type＝3　3d 包络盒由 ebox 参数设定，与 type＝1 类似。

type＝4　3d 包络盒由给定的数据设定，与 type＝2 类似。

type＝5　3d 扩展包络盒由 ebox 参数设定，与 type＝1 类似。

type＝6　3d 扩展包络盒由给定的数据设定，与 type＝2 类似。

② box：　三维图形周围的包络盒参数。

box＝0　围绕图形的包络盒不画。

box＝1　与 box＝0 相同。

box＝2　仅画曲面后面的轴。

box＝3　绘制包围曲面的包络盒，添加每个轴的标题。

box＝4　绘制包围曲面的包络盒，添加轴和标题。

参数 ebox 是当 Flag 中的 type 标记为 1 时使用。用向量

[xmin, xmax, ymin, ymax, zmin, zmax]指定绘制图形的边界。

例 5.15 param3d 绘图示例(见图 5.14)

```
t=0 : 0.1 : 5 * %pi;
param3d(sin(t),cos(t),t/10,35,45,"X@Y@Z",[2,3])
```

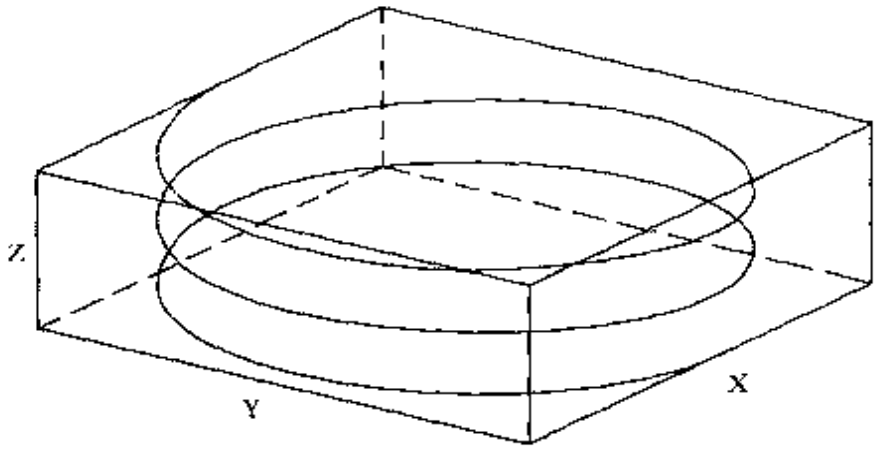


图 5.14 param3d 绘图示例

2. param3d1 函数

函数 param3d1 与函数 param3d 类似,也是一个用于绘制空间参数曲线的函数,其基本调用格式如下:

```
param3d1(x,y,z,[theta,alpha,leg,flag,ebox])
param3d1(x,y,list(z,colors),[theta,alpha,leg,flag,ebox])
```

其中,x,y,z 为具有相同大小 nl×nc 的矩阵。矩阵的列 i 对应第 i 条曲线。可以用表 **list(z, colors)**指定每条曲线的颜色,而不是用 z,这里颜色是一个大小为 nc 的向量。如果 color(i)是负的,则用 id 为 abs(style(i))+1 的标识(mark)绘制图形。

例 5.16 param3d1 绘图示例(见图 5.15)

```
t=[0 : 0.1 : 5 * %pi]';
param3d1([sin(t),sin(2 * t)],[cos(t),cos(2 * t)],…
```

```
list([t/10,sin(t)],[3,2]),35,45,"X@Y@Z",[2,3])
```

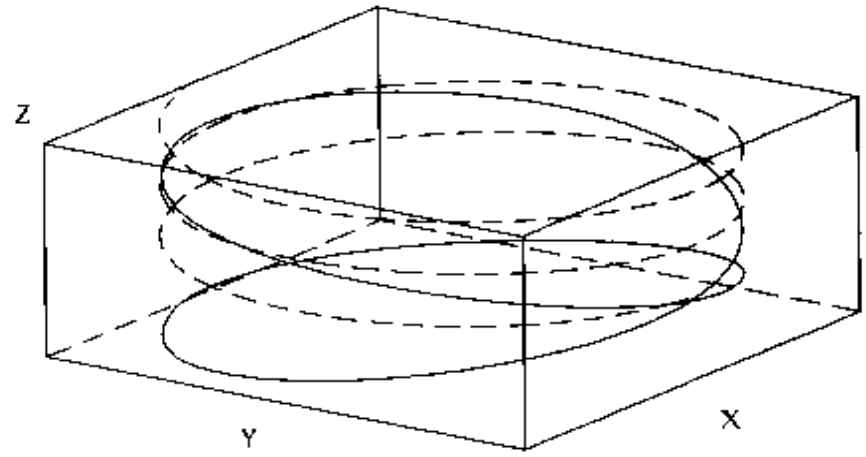


图 5.15 param3d1 绘图示例

5.4 绘图全局参数的设定

如前所述，一些绘图属性可以用绘图指令的自变量来控制；另外一些绘图属性，可以通过设置绘图全局参数来设定。绘图全局参数可以利用指令 xset()来设定。如果在 SCILAB 的命令窗口直接输入指令 xset()，则会打开一个交互式的图形参数设置面板，用户可以通过简单的鼠标点击改变参数，也可以通过带有参数的 xset 指令控制不同的绘图全局参数。

下面介绍用 xset 函数设置绘图全局参数时的基本调用格式：

xset(choice-name,x1,x2,x3,x4,x5)

其中：

choice-name 是一个字符串变量，用于确定所设参数的类型；

x1，…，x5 由具体的 choice-name 决定。

下面是一些常用的 xset 指令：

(1) **xset("background",color)**

设置当前窗口的背景值。

(2) **xset("color",value)**

设置用于填充线段或文本绘制函数等的缺省颜色，value 是一个整数值，取值范围在[0，whiteid]之间。0 用于黑色填充，whiteid 用于白色填充。Whiteid 的值可通过**xget("white")**得到。

(3) **xset("default")**

返回缺省值。

(4) **xset("font size",fontsize)**

设置字体的尺寸。

(5) **xset("foreground",color)**

设置当前图形窗口的前景色。

(6) **xset("thickness",value)**

设置用像素表示的线型的宽度(0 和 1 有相同的意义，都是一个像素宽)。

(7) **xset("viewport",x,y)**

设置视点位置。

(8) **xset("window",window-number)**

设置当前窗口的窗口号，如果这个窗口不存在，则建立新的窗口。

(9) **xset("wpos",x,y)**

设置当前图形窗口的左上点位置。

(10) **xset("wdim",width,height)**

设置当前图形窗口的宽度和高度。

(11) **xset("pattern",value)**

设置填充函数的当前模式，value 是一个取值范围为[0，whiteid]的整数值。

(12) **xset("line style",value)**

设置当前线型(1：实心，>1：虚线)。

(13) **xset("clipping",x,y,w,h)**

设置裁剪区域为长方形(x,y,w,h)(左上点的长宽)(在这个图形窗口中可以绘制图形),这个函数用当前的坐标绘制。

(14) **xset("wresize",flag)**

如果 flag=1,图形自动根据图形窗口的尺寸调节大小。

(15) **xset("font",fontid,fontsize)**

设置当前字体的类型和字体的大小。

(16) **xset("auto clear","on"|"off")**

利用开关"on"或"off"设置图形的自动清除模式。当自动清除模式设为"on"时,连续的绘图不会重叠,就像每次绘图时都执行一次 **xbasc()**。缺省值是"off"。

(17) **xset("colormap",cmap)**

设置色图(colormap)为一个 m×3 的矩阵,m 种颜色。颜色 i 分别由对应着红、绿和蓝三基色的 cmap(i,1), cmap(i,2)和 cmap(i,3)给定,取值范围在 0～1 之间。

(18) **xset("mark",markid,marksize)**

设置当前绘图标识(mark)及其大小。利用 getmark()函数可以得到所有标识类型,如图 5.16 所示。注意设置的标识尺寸 marksize 不仅对本标识 markid 有效,而且对所有的标识都有效。

(19) **xset("mark size",marksize)**

设置当前标识的尺寸。

(20) **xset("hidden3d",colorid)**

设置 plot3d 函数中隐藏面的颜色索引值 colorid。如果 colorid=0,则不绘制三维图形的隐藏面。

例 5.17 打开多个图形窗口绘图

```
xset("window",1);
plot3d();
xset("window",2);
```

```
plot3d();
```

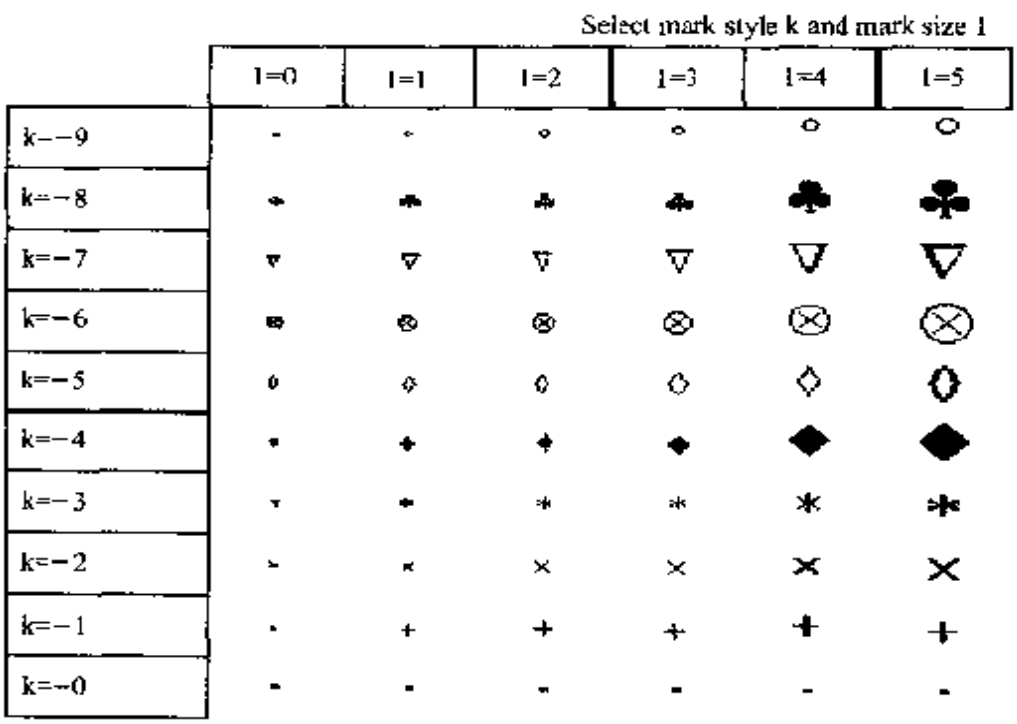


图 5.16 绘图标识类型

例 5.18 利用不同的线型绘图(见图 5.17 和图 5.18)

```
xset("line style",1)
xset("thickness",1)
x=[0:0.1:2*%pi];
plot2d(x, sin(2*x))
xbasc();
xset("line style",2)
x=[0:0.1:2*%pi];
plot2d(x, sin(2*x))
```

说明:指令 xset("line style",1)将绘图的线型设为实线;xset("thickness",1)将线的宽度设为一个像素,在这种情况下,生成如图 5.17 所示的图形;xset("line style",2)将绘图的线型设为虚

线,生成的图形如图 5.18 所示。

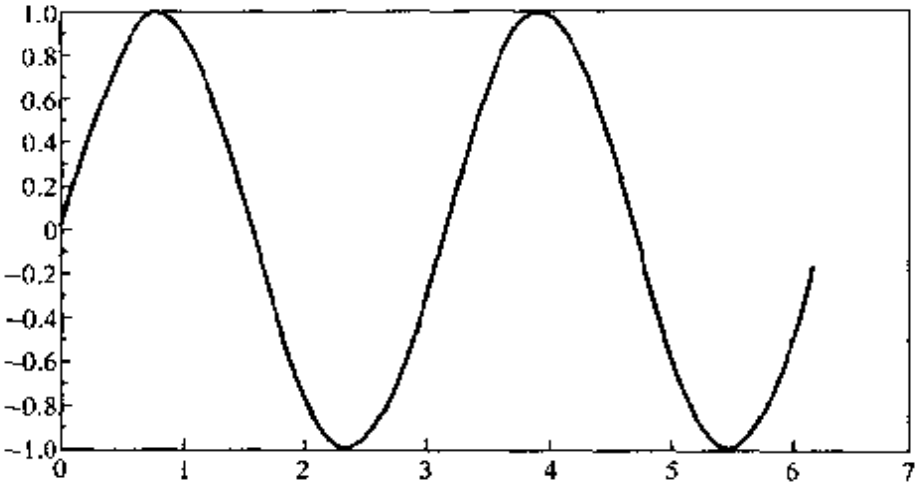


图 5.17 xset 的应用(实线)

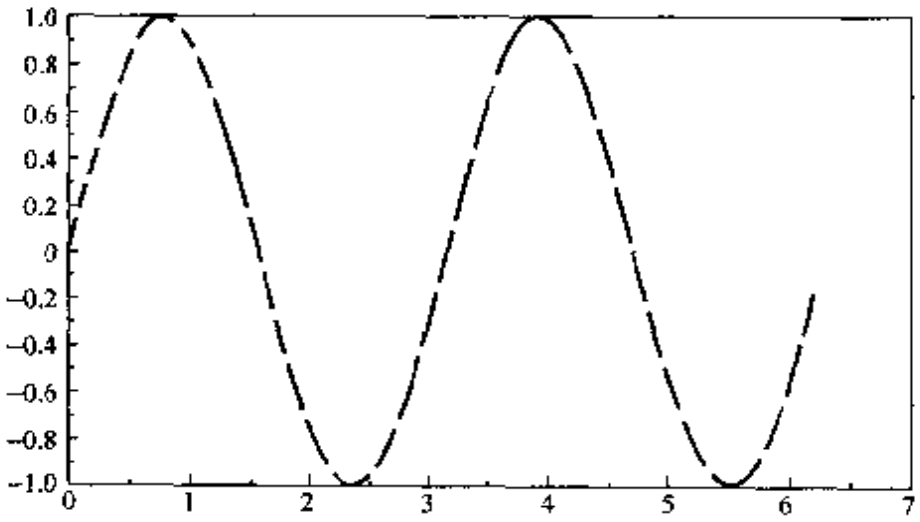


图 5.18 xset 的应用(虚线)

例 5.19 利用不同的标识绘图(见图 5.19)

```
xset("mark size",5);
xmax=5;x=0:0.1:xmax;
u=[-0.8+cos(x);-0.6+cos(x);-0.4+cos(x);-0.2+cos(x);
   cos(x)];
u=[u;0.2+cos(x);0.4+cos(x);0.6+cos(x);0.8+cos(x)]';
```

原书缺页

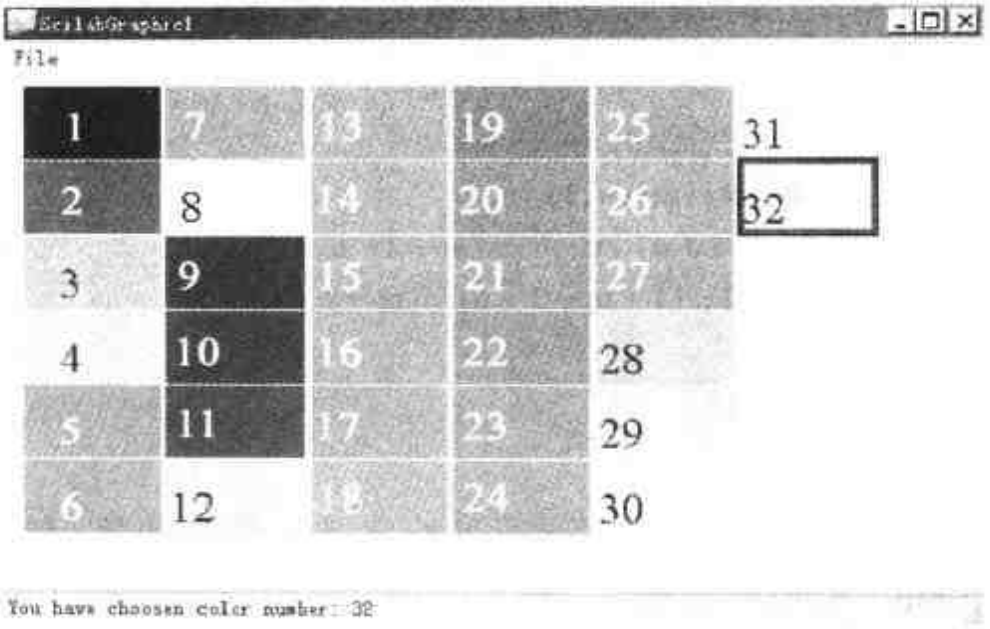


图 5.20 用 getcolor()指令得到的当前色图

表 5.1 当前色图所对应各颜色的名称

色图号	颜色名称	色图号	颜色名称	色图号	颜色名称
0	黑色	11	深蓝色	22	深紫色
1	黑色	12	天空蓝	23	鲜紫色
2	深蓝色	13	深绿色	24	深棕红色
3	淡绿色	14	深绿色	25	深棕红色
4	天空蓝	15-	鲜绿色	26	棕红色
5	鲜红色	16	深绿色	27	深橙色
6	紫色	17	深蓝绿色	28	粉红色
7	鲜红色	18	蓝绿色	29	粉红色
8	白色	19	深红色	30	粉红色
9	淡蓝色	20	深红色	31	粉红色
10	蓝色	21	红色	32	深黄色

例 5.20 通过设置色图，改变所绘图形的颜色。(见图 5.21)

```
m=228;
n=fix(3/8 * m);
r=[(1 : n)'/n;ones(m-n,1)];
g=[zeros(n,1); (1 : n)'/n; ones(m-2 * n,1)];
b=[zeros(2 * n,1); (1 : m-2 * n)'/(m-2 * n)];
h=[r g b];
xset("colormap",h)
plot3d1()
```

说明：fix 函数用于取整。通过给定 r，g 和 b 的值，利用 **xset("colormap",h)**设定色图。

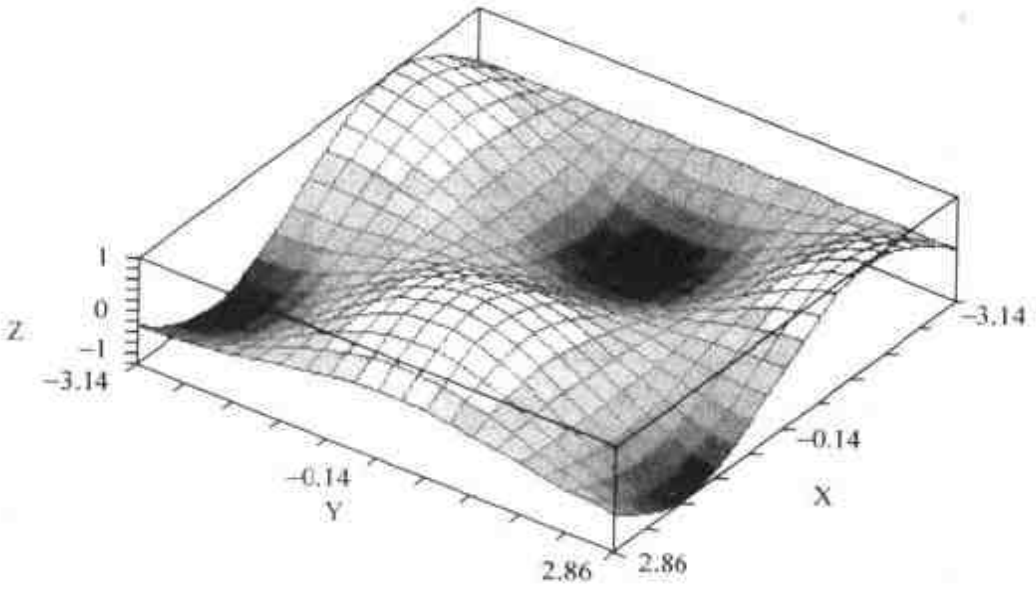


图 5.21 利用色图调制的图形

SCILAB与C或FORTRAN程序的接口

SCILAB提供与C语言和FORTRAN语言的接口，可以用C语言和FORTRAN程序扩展SCILAB的功能。SCILAB最简单的调用外部程序的方法是用link这个基本指令。这个指令可以动态地将用户的程序和SCILAB链接。然后用call这个基本指令调用所链接的程序。另一个添加C或FORTRAN代码到SCILAB的方法是通过建立一个接口程序。这个接口程序可以按照目录SCILAB/routines/examples/interface-tutorial和routines/examples/interface-tour中的例子写。在目录SCILAB/routines/examples/mexfiles中有与MATLAB类似的接口例子。接口程序也能用intersci这个程序建立。Intersci从一个.desc文件建立。通过建立一个接口程序，可以给SCILAB添加新的基本命令(需刷新fundef文件)，或建立一段新的可执行代码。

6.1 应用动态链接

在目录examples/link-examples中给出了几个简单的动态链接例子。本节简单介绍怎样调用动态链接程序。

6.1.1 动态链接

指令 link()用于将编译过的目标文件链接到 SCILAB 中。例如 **link('path/pgm.o','pgm',flag)** 将路径 path 下的目标文件 pgm.o 链接到 SCILAB。这里 pgm 是一个调用名(entry point),即在 SCILAB 中可以通过这个名字调用所链接的程序。一个目标文件可能有几个调用名,可以用例如['pgm1','pgm2']的字符串向量添加多个调用名。对于 C 程序,flag 应设为'C';对于 FORTRAN,flag 应设为'F'(缺省设置)。如果链接成功,则 SCILAB 返回一个与链接程序相关的整数 n。Ulink(n)命令用于返回没有链接前的状态。如果目标文件链接到了 SCILAB,则命令 **c_link('pgm')** 返回真,否则返回假。

例 6.1 链接 FORTRAN 程序

```
-->n=link(SCI+'/routines/blas/daxpy.o','daxpy')
-->n =
        0.
-->c_link('daxpy')
        ans =
            T
-->ulink(n)
-->c_link('daxpy')
        ans =
        F
```

说明:这里所要链接的文件是用 FORTRAN 语言编成的目标文件 daxpy.o,调用名设为 daxpy。链接成功返回的整数 n 的值是 0。如果链接成功,用 **c_link('daxpy')** 测试,返回值为真(T)。用 ulink(n)返回到没有链接状态,然后再用 **c_link('daxpy')** 测试,这时由于没有链接,所以返回值为假(F)。

6.1.2 调用动态链接程序

在 SCILAB 中,可用 call 函数调用一个动态链接程序。使用 call 函数的格式有两种:

(1) **[y1,…,yk]=call("ident",x1,px1,"tx1",…,xn,**
pxn,"txn","out",[ny1,my1],py1,
"ty1",…,[nyl,myl],pyl,"tyl")

(2) **[y1,…,yk]=call("ident",x1,…,xn)**

说明:在上面两种调用格式中,短调用格式(2)不仅语法简单,且能生成较快的运行代码,但用户必须写一个小接口程序;长调用格式(1)虽然语法复杂,但不需要接口程序。

每个参数的意义如下:

ident:所调用程序的名字。

x1,…,xn:传递到程序的输入变量,是实数矩阵或字符串类型。

px1,…,pxn:所调用程序中各变量的位置。

tx1,…,txn:各变量的类型,分别用符号"r","i","d"和"c"代表实数(浮点数)、整数、双浮点数和字符串类型。

out:是一个用于分隔输入变量和输出变量的关键词。当这个关键词出现时,表明使用的是 call 的长调用格式,否则为短调用格式。

[ny1, my1]:输出变量的大小。

py1:输出变量的位置。

ty1:是输出变量在原 FORTRAN 或 C 语言中的类型。

下面是采用 call 函数长调用格式的一个例子。

例 6.2 call 函数的长调用格式

```
void fooc(c,a,b,m,n)
```

```
  double a[], * b,c[];
int  * m, * n;
{
  double sin();
  int i;
  for ( i =0 ; i < ( * m) * ( * n) ; i++)
    c[i] = sin(a[i]) +  * b;
}

-->link("fooc.o","fooc","C")
-->a=[1,2,3;4,5,6];
-->b= %pi;
-->[m,n]=size(a);
-->c=call("fooc",a,2,"d",b,3,"d",m,4,"i",n,5,…
"i","out",[m,n],1,"d");
```

说明：用SCILAB执行C或FORTRAN程序时，它的输入参数必须是从SCILAB转换来的特定值，输出参数必须转换成SCILAB变量。本例中将一个用C语言写的程序调用到SCILAB中，因此在用link函数链接时，要用符号"C"指明。具体应用时，首先在C语言环境下将fooc函数编译成目标文件fooc.o；然后在SCILAB中用link函数链接。本例中的调用名给定为fooc。这样在SCILAB中就可以用call函数调用C语言写的fooc程序了。

6.2 接口程序

在SCILAB中调用较复杂的用C或FORTRAN语言写的程序，一般需要建立接口程序。由于用SCILAB执行C或FORTRAN程序时，其输入参数必须符合SCILAB的变量格式，

输出参数也必须转换成 SCILAB 变量，因此用接口程序完成这部分工作，使在 SCILAB 中调用 C 或 FORTRAN 程序时，自动完成输入输出数据的格式转换工作。

6.2.1 建立一个接口程序

在目录 scidir/examples/interface-tutorial 和 scidir/examples/interface-tour 中给出了接口程序的例子。接口程序由 C 或 FORTRAN 语言编写。注意一个单独的接口程序可用于接口任意多的函数(但小于 99 个)。

下面通过一个例子来说明接口程序的编写和使用方法。

例 6.3 将一个用 C 语言编写的矩阵乘法程序接到 SCILAB 的接口程序。

```
/* Matrix multiplication C=A*B, (A,B,C stored columnwise) */
#define A(i,k) a[i + k * n]
#define B(k,j) b[k + j * m]
#define C(i,j) c[i + j * n]
void matmul(a,n,m,b,l,c)
double a[],b[],c[];
int n,m,l;
{
  int i,j,k; double s;
  for(i=0 ; i<n; i++)
  {
    for(j=0; j<l; j++)
    {
      s = 0.;
      for(k=0; k<m; k++)
      {
        s += A(i,k) * B(k,j);
      }
```

```
            C(i,j) = s;
          }
        }
      }
```

说明：以上的C语言函数matmul的功能是使两个矩阵A和B相乘，返回矩阵C。本例的目的是在SCILAB中也能使用这个函数(函数名也称matmul)，例如：

```
-->C=matmul(A,B)
```

由于是在SCILAB中调用，所以这里A，B和C是标准的SCILAB矩阵。在调用时，SCILAB矩阵A和B应该能被传递到C函数matmul中，矩阵C在C函数matmul中被建立添充后，传递回SCILAB中。欲建立SCILAB函数matmul，必须写如下的接口（gateway）程序，程序命令为intmatmul。接口程序intmatmul也需要用C语言书写：

```
#include "stack-c.h"
int intmatmul(fname)
char *fname;
{
  static int l1, m1, n1, l2, m2, n2, l3;
  static int minlhs=1, maxlhs=1, minrhs=2, maxrhs=2;
  CheckRhs(minrhs,maxrhs); CheckLhs(minlhs,maxlhs);
  /* Get A (#1) and B (#2) as double ("d") */
  GetRhsVar(1, "d", &m1, &n1, &l1);
  GetRhsVar(2, "d", &m2, &n2, &l2);
  /* Check dimensions */
  if(!(n1==m2))
  {
    Scierror(999,"%s: Uncompatible dimensions\r\n",fname);
    return 0;
  }
```

```
  /* Create C (#3) as double ("d") with m1 rows and n1 columns */
  CreateVar(3, "d", &m1, &n2, &l3);
  /* Call the multiplication function matmul inputs:stk(l1)->A, stk
(l2)->B output:stk(l3)->C */
  matmul(stk(l1), m1, n1, stk(l2), n2, stk(l3));
  /* Return C (3) */
  LhsVar(1) = 3;
  return 0;
}
```

说明：如程序第一行所示，接口程序必须包含头文件 SCIDIR/routines/stack-c.h。这个头文件是 SCILAB 专为接口程序所写的头文件，定义了一些编写接口程序所需的特定函数。这个接口程序命名为 intmatmul，它允许有一个输入参数 fname。fname 必须声明成 char *。接口程序的名字(这里是 intmatmul)是随意的，但是参数 fname 是必需的。接口中需包含所用到的 C 变量的说明。

函数 CheckRhs 和 CheckLhs 分别对应 SCILAB 调用函数的输入与输出变量个数的设定，设定 matmul 函数输入(右边，Rhs)和输出(左边，Lhs)参数个数的范围。由于在 SCILAB 中将会以 C=matmul(A,B)的形式使用，所以函数的输入参数是 2 个，输出参数是 1 个。所以这里 minlhs = 1，maxlhs = 1，minrhs = 2，maxrhs=2。例如，当输入 matmul(A)时，将会给出由 CheckRhs 生成的错误信息。

在 SCILAB 中，所用到的参数自动按先后顺序编号为 1，2，3，…。例如在本接口程序中，矩阵 A，B 和 C 分别用数字 1，2 和 3 代表。建立的 SCILAB 函数 matmul 的每个输入变量通过调用 GetRhsVar 函数来处理。GetRhsVar 函数的前 2 个参数是输入参数，后 3 个参数是输出参数。例如 GetRhsVar(1, "d", &m1, &n1, &l1)用于处理输入(Rhs)变量。它的第一个参数(这里是

1)对应 SCILAB 函数 matmul 的第一个参数(这里是 A)。GetRhsVar 的第 2 个参数(这里是"d")表示变量的类型,A 是一个 SCILAB 数字矩阵,它的类型是"d",即双浮点型。因为在 C 程序,矩阵 A 要求是一个双浮点型矩阵。从对这个 GetRhsVar 的调用中,可以知道 A 有 m1 行和 n1 列。同理,GetRhsVar(2, "d", &m2, &n2, &l2)对矩阵 B 作了设定。

矩阵 A 的列的数目应该等于矩阵 B 的行的数目。当传输到 matmul 的矩阵 A 和 B 的维数不匹配时,程序将停止运行,在 SCILAB 中返回错误信息:"Uncompatible dimensions"。

函数 CreateVar(3, "d", &m1, &n2, &l3)用于建立接输出变量 C。变量 C 是一个 m1 行 n2 列的整型矩阵。如前所述,编号 3 代表矩阵 C。CreateVar 的调用序列与 GetRhsVar 的调用序列相同,但前 4 个参数是输入。通过对 CreateVar 和 GetRhsVar 的调用,得到 SCILAB 中输入的变量 A,B 和 C 的变量地址:stk(l1)->A,stk(l2)->B 和 stk(l3)->C。

接下来调用原 C 语言函数 matmul。原函数定义如下:

```
void matmul(a,n,m,b,l,c)
double a[],b[],c[]; int n,m,l;
```

调用时输入参数必须按照这个顺序。必须发送指向矩阵 A,B 和 C 地址的指针 stk(l1),stk(l2)和 stk(l3)。例如本例中 matmul(stk(l1), m1, n1, stk(l2), n2, stk(l3))。这里 stk(l1)是指向矩阵 A 的指针,A 中的单元以 stk(l1)[0], stk(l1)[1]等方式存储。类似地,调用完 C 语言函数 matmul 后,stk(l3)[0], stk(l3)[1]存储 A * B 的结果。

在程序的最后需将结果返回到 SCILAB 中,例如这里将矩阵 C 返回到 SCILAB。由于矩阵 C 的编号为 3,所以用 LhsVar(1)=3 实现返回。接口程序写好后,应该进行编译和 SCILAB 链接,在

SCILAB 中执行脚本文件用于加载新函数。

6.2.2 编译接口程序

建立完接口程序后，为了能够与 SCILAB 链接，需要将其编译成目标文件。在 Windows 操作系统下，可用 Microsoft Visual C++ 将接口程序intmatmul. c编译成目标文件 intmatmul. obj，为了使用方便，可以将其改名为 intmatmul. o。具体地可使用VC++的Debug 菜单下的 Compile 命令，如图 6.1 所示。

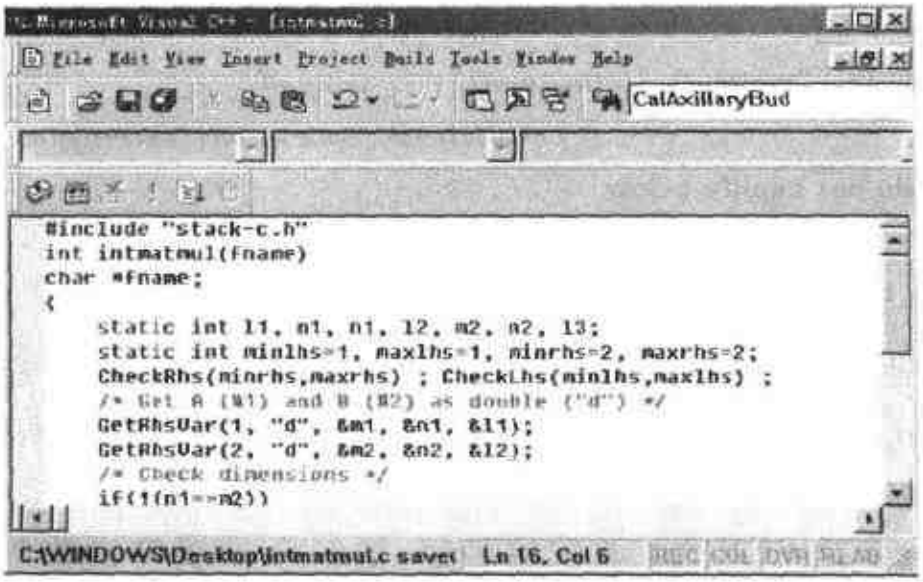


图 6.1 编译接口程序

6.3 建立动态链接库

在目录 SCIDIR/examples/interface-tutorial-so 下包含建立动态库(在 Windows 下是 dll 文件)的必要材料。动态库能动态地与 SCILAB 进行链接。欲建立动态链接库，必须首先生成一个脚本文件 builder. sce。这个文件可以自己编写，也可以直接复制已

有的文件，然后根据用户所需修改这个文件，生成合适的新builder.sce文件。

builder.sce文件的具体内容如下：

```
// 这是 builder.sce示例文件
// 此文件必须在当前目录下运行
ilib_name = "libtutorial" // interface library
                          //name
files = ["intmatmul.o"] // objects
                          files
libs = []                 // other libs needed for linking
table = ["matmul","intmatmul"]; // table of
                          //(SCILAB_name,interf//ace-name)
// do not modify below
// ----------------------------
ilib_build(ilib_name,table,files,libs)
```

这是一个SCILAB脚本文件，注意变量files(字符串行向量)和table(一个具有两列的字符串矩阵)。files应该包含所有目标文件的名字(接口函数和C函数)。table的每一列是两个字符串对：第一个是SCILAB函数的名字；第二个是接口函数的名字。这里函数是matmul，其接口函数是intmatmul。在文件builder.sce被编译后，在SCILAB中应该能执行如下命令：

```
-)exec ('builder.sce')
```

运行这个命令，将会得到以下结果：

```
ilib_name  =
libtutorial
 files  =
!  intmatmul.o!
 libs  =
```

```
[]
generate a gateway file
generate a loader file
generate a Makefile: Makelib
running the makefile
```

通过在 SCILAB 中运行上述命令，将产生文件 Makelib. mak 和文件 loader. sce。其中 Makelib. mak 文件的内容如下：

```
# generated by builder. sce: Please do not edit this file
# ----------------------------------------------------------
SHELL = /bin/sh
SCIDIR =D:/SCILAB
SCIDIR1 =D:\SCILAB
# name of the dll to be built
LIBRARY = libtutorial
# list of objects file
OBJS = matmul. obj intmatmul. obj libtutorial. obj # added libraries
OTHERLIBS =
##### do not edit below this line ##########
! include $ (SCIDIR1)\config\Makedll. incl
```

需要利用 Makelib. mak 文件生成动态链接库 libtutorial. dll。在 Windows 操作系统下，可用 Microsoft Visual C++ 的编译环境编译Makelib. mak文件，生成动态链接库 libtutorial. dll，如图 6. 2 所示。

loader. sce 文件的内容如下：

```
// generated by builder. sce
libtutorial_path=get_file_path('loader. sce');
functions=['matmul';];
addinter(libtutorial_path+'/libtutorial. dll','libtutorial',functions);
```

说明：接口程序写好后，需将其编译成一个目标文件，然后用

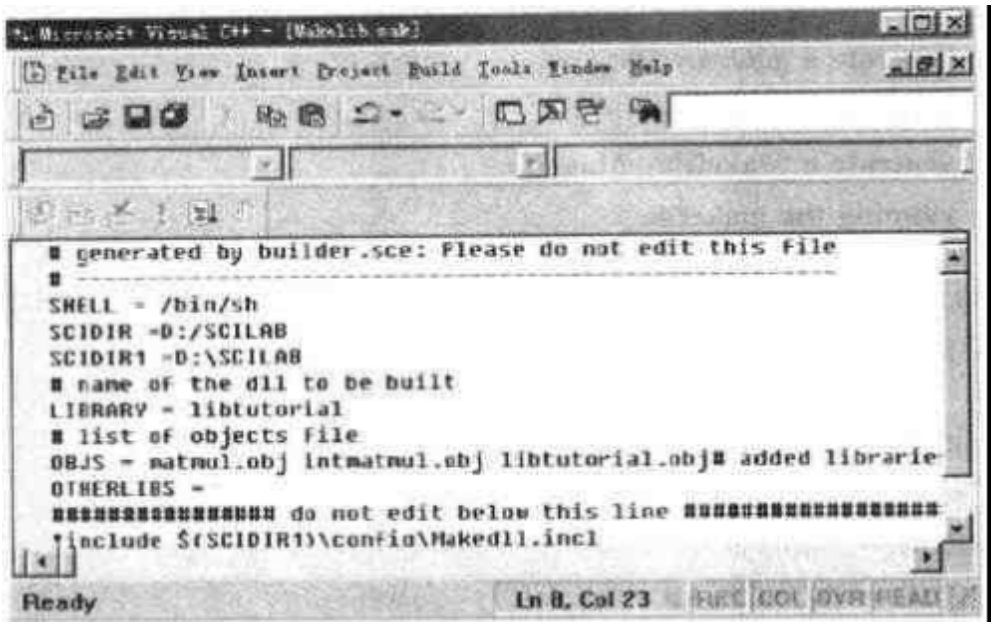


图 6.2 编译 Makelib.mak 文件生成的动态链接库

addinter 指令链接 SCILAB。addinter 指令的格式如下：

addinter(files,spname,fcts)

其中：

files：在 SCILAB 中需要调用的动态链接库名；

spname：在 SCILAB 中调用接口函数名(entry point)；

fcts：在新的接口函数中应用的新的 SCILAB 函数名。

在 SCILAB 中，通过执行 loader.sce 文件装载已经建立的指令 matmul。

```
-->exec ('loader.sce')
-->A=rand(2,3);B=rand(3,3);C=matmul(A,B); //C=A * B
```

小结：为了建立一个动态接口，用户必须首先写一个接口函数(例如 intmatmu)；然后编译这个接口函数，生成接口函数的目标文件(如 intmatmu.o)；接着编辑 builder.sce 文件。可以直接复制一个现成的文件，然后修改它，也即输入将建立的动态链接库

名、所要链接的目标文件名、所要建立的 SCILAB 的函数名和接口函数名。通过在 SCILAB 中执行脚本文件 builder. sce，产生 Makelib. mak 文件和脚本文件 loader. sce。接着编译 Makelib. mak 文件，生成 SCILAB 的动态链接库。当你每次需要应用新的函数时，需先执行脚本文件 loader. sce，然后用和 SCILAB 的基本函数一样的格式调用新函数即可。

第 7 章

SCILAB 应用篇

7.1 信号处理

SCILAB 目前提供了信号处理工具箱。但是本书主要侧重于最为基本的指令的应用，并以讲解实例为主。

例 7.1 生成白噪音信号

信号处理中经常会用到随机信号的生成。SCILAB 提供的标准随机信号生成指令为

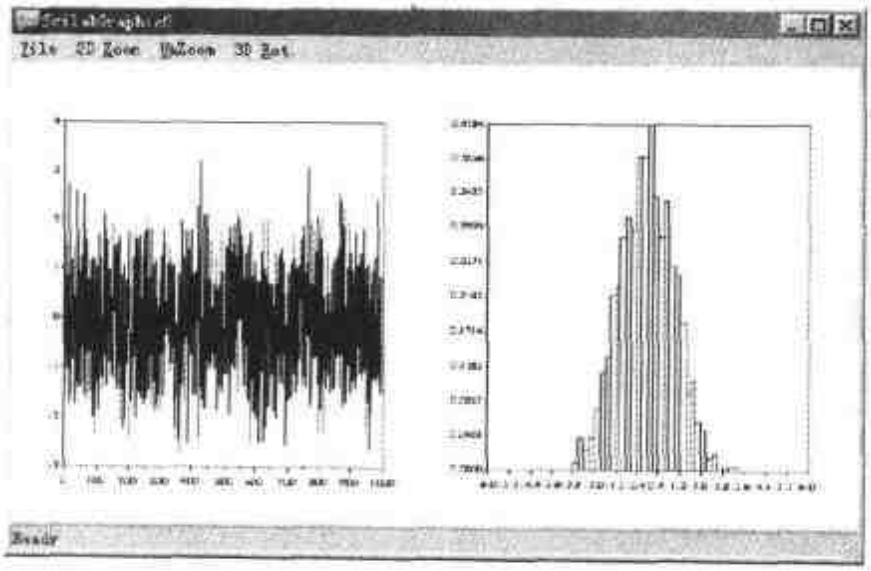
```
x=rand(m1,m2,… [,'string'])
```

其中，m1，m2，…代表维数形式：m1 × m2 × …；'string' = 'uniform'将生成[0,1]之间均匀分布的随机信号；'string' = 'norma'将生成标准高斯分布随机信号，其均值为零，方差为 1。

下面是白噪音时间序列生成语句。该序列符合标准高斯分布。下面是应用实例：

```
rand('normal');    // 初始设置为标准高斯分布
rand('seed',0);    // 设置随机生成器的初始"种子数"为 0(可以是任
                   // 意实数)
```

```
x=rand(1:512); // 生成一维白噪音(标准高斯分布)时间序列
xsetech([0.0,0.0,0.5,1.0]); plot(x)   // 见图 7.1(a),为白噪音(标
                                      // 准高斯分布)时间序列
xsetech([0.5,0.0,0.5,1.0]); histplot([-6:0.2:6],x,[1,-1],'011');
                                      // 见图 7.1(b),为时间序列
                                      // 的直方图显示
```



(a) 标准高斯分布白噪音时间序列 (b) 时间序列的直方图

图 7.1 白噪音信号显示

下面是另一个例子:

```
-->x=rand(2,4,2,'uniform')
x  =
(:,:,1,1)
!    .9922061    .4623708    .6244548    .3693783 !
!    .4348401    .8428979    .9890787    .3564726 !
(:,:,2,1)
!    .4829412    .9864565    .5199562    .2696768 !
!    .7250442    .0797251    .8990914    .2452792 !
```

例 7.2 FFT及功率谱计算

功率谱计算在信号分析中是常规的操作。在此将使用FFT(快速傅里叶变换),该指令的基本格式如下:

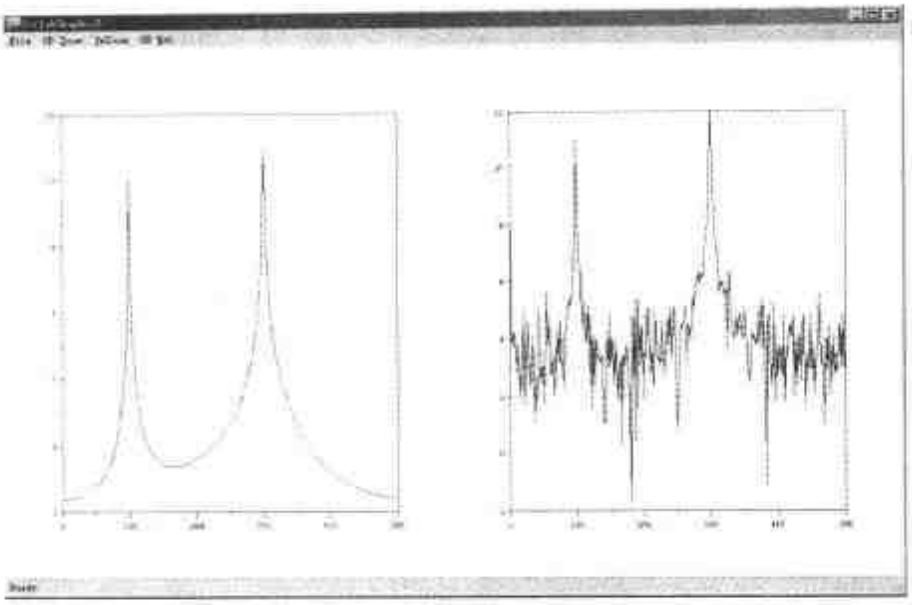
```
x=fft(a,i,dim,incr)
```

其中,a是将变换的输入数据;i = −1为直接快速傅里叶变换,i=1为逆变换;dim与incr是在对多维输入数据进行处理时使用。此例是已知有多路正弦信号受有加性噪音的污染,预估正弦信号的最大频率不超过500Hz。根据Nyquest的采样定理(至少要大于两倍的感兴趣频率),选择采样频率为1000Hz。因此采样间隔时间为0.001s,采样总个数设定为512。下面是对源信号x(t)与有加性噪音污染信号进行的功率谱计算及显示。注意,经FFT计算的数据是对称中心的数据,因此只显示前一半数据即可以完全表述计算结果。

```
Fs=1000; n=512;                    // 给定的采样频率与采样总个数
dt=1/Fs; t=0:dt:dt*(n-1);          // 采样间隔计算与生成时间序列
x=sin(2*%pi*100*t)+2*cos(2*%pi*300*t);   // 源信号
y=rand(1:n)+x;                     // 加性噪音污染信号
z=fft(y,-1); u=fft(x,-1);          // FFT计算,FFT后的输出一般
                                   // 为复向量
f=1000*(0:n/2-1)/n;                // 频率序列()
Pyy=z.*conj(z); Pxx=u.*conj(u);    // 功率谱计算
xsetech([0.0,0.0,0.5,1.0]); plot2d(f', log(Pxx(1:256)))
      // 图 7.2(a) 源信号功率谱显示
xsetech([0.5,0.0,0.5,1.0]); plot2d(f', log(Pyy(1:256)))
      // 图 7.2(b) 加性噪音污染信号功率谱显示
```

例 7.3 FIR低通滤波器设计

有关线性FIR(有限脉冲响应)式数字滤波器的基本指令如下:



(a) 源信号功率谱图 (b) 加性噪音污染的信号功率谱图

图 7.2 功率谱图显示

[wft,wfm,fr]=wfir(ftype,forder,cfreq,wtype,fpar)

其中,ftype 设定滤波器类型,包括:'lp','hp','bp','sb',分别为低通、高通、带通及带阻;forder 是滤波器阶次; cfreq=[low, high]是低高通带截止频率,规定在[0, 0.5]之间; wtype 设定加窗函数类型,包括:'re','tr','hm','hn','kr'和'ch',分别为矩形窗、三角函数窗、Hamming 窗、Hanning 窗、Kaiser 窗和 Chebyshev 窗; fpar 是窗函数参数设定;wft 是滤波器设计参数输出; wfm 是频域的幅值响应输出,fr 是[0, 0.5]之间的频域序列。当在 SCILAB 指令窗口下直接键入 wfir()时,SCILAB 将自动弹出对话窗口完成输入。下面是应用指令完成的低通数字滤波器设计示例。注意,图 7.3 中的横坐标是单位化的频率范围,纵坐标是分贝(dB)量纲。在 −3dB 的幅值响应输出点上基本对应截止频率为 0.3。

```
[wft,wfm,fr]=wfir('lp',50,[0.3 0],'hm',[ ]);  // 低通数字滤波
                                               // 器计算
```

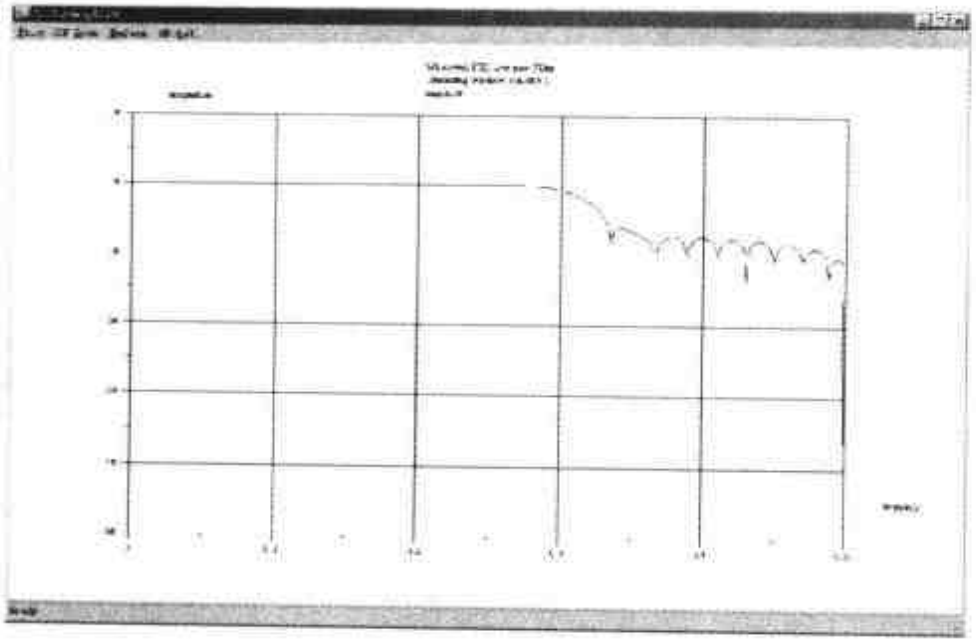


图 7.3 低通数字滤波器幅值响应

```
plot2d(fr',log(wfm)')                          // 图 7.3 幅值响
                                               // 应特性显示
xtitle(' ','frequency','magnitude');           // 文字注释
xtitle(['Windowed FIR Low pass Filter';...
    ' Hamming window, cut-off : 0.3'; 'length 50'])
```

7.2 系统分析

例 7.4 "质量-弹簧-阻尼"系统动态时域响应

这是一个一维自由度的"质量-弹簧-阻尼"系统(见图 7.4)。它的传递函数是如下形式:

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega^2 s + \omega_n^2}$$

其中,ω_n与ζ分别为系统的固有频率和阻尼。这是单环路二阶系统,可以使用 SCILAB 中有关线性系统响应指令:

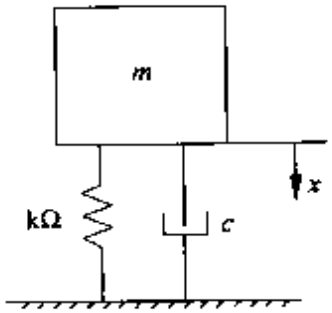
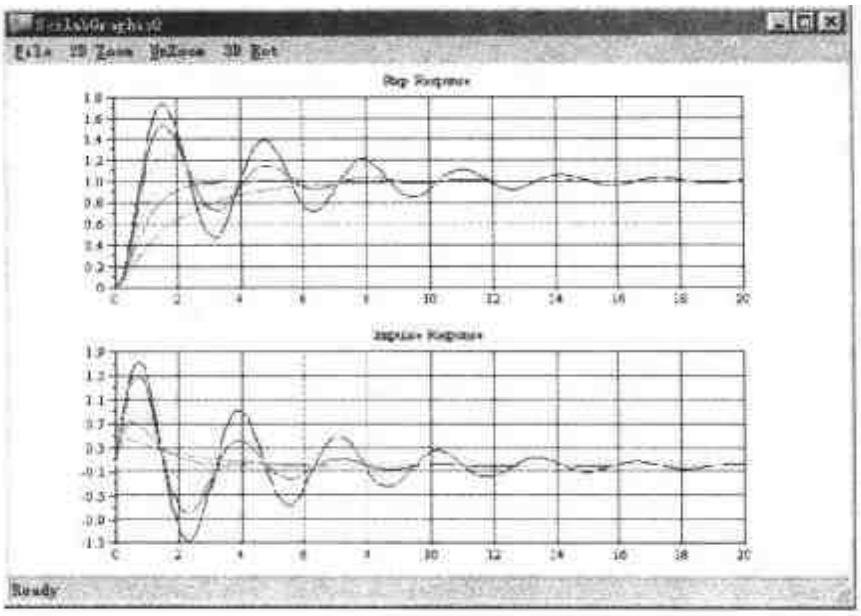


图 7.4 一维自由度的"质量-弹簧-阻尼"系统

```
[y [,x]]=csim(u,t,sl[,x0 [,tol]])
```

其中，u 表示激励响应类型，包括："step"与"impulse"，分别为单位阶跃和单位脉冲激励类型；t 是时间序列向量；sl 是线性系统的 list 形式表达式；x0 为系统激励信号初始值(x0＝x(t(1))；tol 是仿真允许误差；x 与 y 分别为系统的激励信号与相应信号矩阵。

在本例中我们希望考察阻尼 ζ 在单位阶跃响应和单位脉冲响应中的影响。应用如下 SCILAB 程序计算后，得到图 7.5。图 7.5



上图：阶跃响应；下图：脉冲响应

图 7.5 "质量-弹簧-阻尼"系统动态时域响应曲线

中的高阻尼ζ值对应的响应曲线波动振幅将更低，且更快地收敛到平衡点。

```
wn=2;instants=0:0.05:20;                    // 设定固有频率,生成时间
                                            // 序列
kexi=[0.1 0.2 0.4 0.7 1.0 2.0];             // 设定阻尼序列
for i=1:length(kexi) // 根据阻尼情况计算响应
    s=poly(0,'s');
    h=wn^2/(s^2+2*kexi(i)*wn*s+wn^2);       // 构造系统传递
                                            // 函数
    sl=syslin('c',h);
    y=csim('step',instants,sl);             // 阶跃响应
    y1(i,1:length(y))=y;
    z= csim('imp',instants,sl);             // 脉冲响应
    z1(i,1:length(z))=z;
end
xbasc();xsetech([0.0,0.0,1.0,0.5]);plot2d1("onn",instants',y1')
xtitle("Step Response"); xgrid(2)
xsetech([0.0,0.5,1.0,0.5]); plot2d1("onn",instants',z1')
xtitle("Impulse Response"); xgrid(2)
```

例7.5 “质量-弹簧-阻尼”系统频域响应

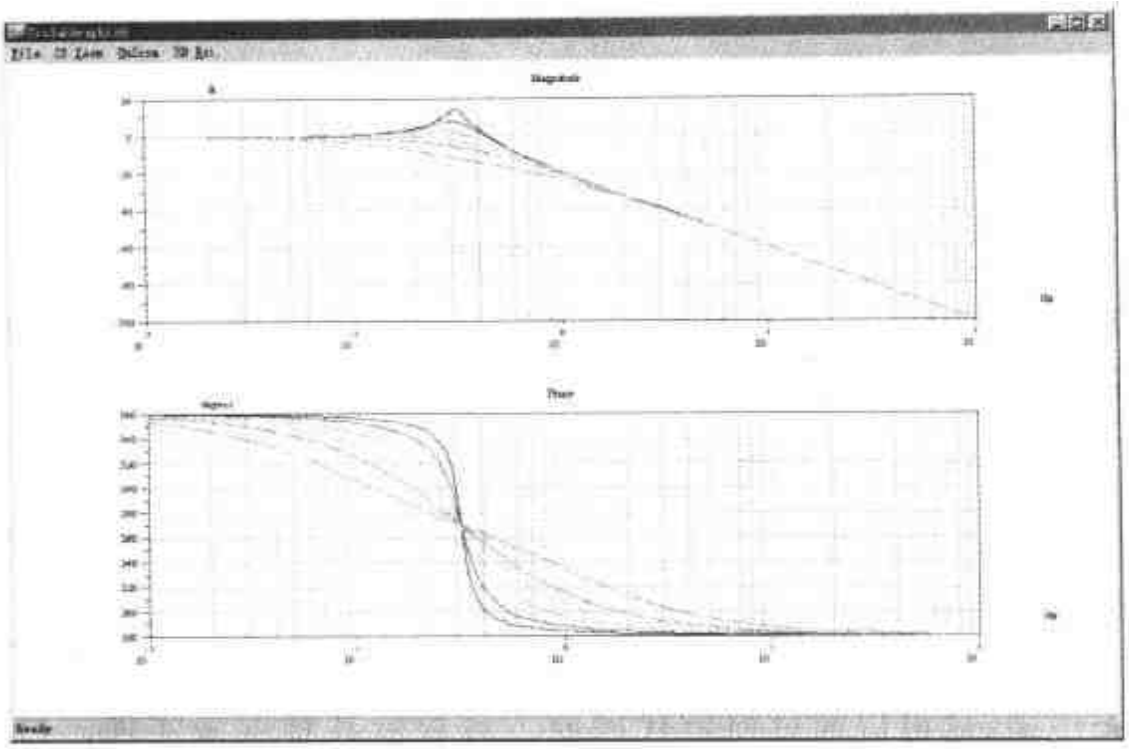
根据上面系统的传递函数，可以应用经典的伯德图形式表现系统的频域响应。其中，SCILAB中的伯德图应用指令如下：

```
bode(sl,fmin,fmax [,comments])
```

其中，sl是线性系统的有理形式表达式；fmin与fmax分别为伯德图显示频率的上下限范围，以赫兹(Hz)为单位；comments可以是字符串向量，用于作图形的文字注释。下面是关于上述例题的有关程序。

```
wn=2;instants=0:0.05:20;      // 设定固有频率,生成时间序列
kexi=[0.1 0.2 0.4 0.7 1.0 2.0]; // 设定阻尼序列
```

```
hh=[];
for i=1 : length(kexi)                    // 根据阻尼情况计算响应
    s=poly(0,'s');
    h=wn^2/(s^2+2*kexi(i)*wn*s+wn^2); //  构造系统传递函数
    h1=syslin('c',h)
    hh=[hh;h1];
end
xbasc(); bode(hh,0.01,100);
```



上图：幅频曲线；下图：相频曲线

图 7.6 “质量-弹簧-阻尼”系统频域响应曲线

图 7.6 中的上图是幅值响应曲线；下图是相位响应曲线。从图中可以观测到，随着阻尼系数的变化，其响应曲线是不同的。当阻尼系数很小时，系统的幅值响应将具有很明显的共振峰。共振峰的所在频率可以由下式计算：

$$f_R = \frac{\omega_d}{2\pi} = \frac{\omega_n\sqrt{1-\zeta^2}}{2\pi}$$

当带入 $\omega_n=2$ 与 $\zeta=0.2$ 后，很容易计算出 $f_R=0.312$Hz。图中峰值所在频率，或相位图中曲线过 180°点位置也基本上证实了这一点。

7.3 线性方程组求解

SCILAB提供若干线性方程组求解指令。其中最常见的是逆矩阵方法：$x=A^{-1}b$，对应指令是 inv，该指令只能用在对满秩矩阵的求解上；另一个指令是用“左除”符号“\”，该指令最为灵活，不仅适用于“超定”矩阵，还适用于“欠定”矩阵，其解是根据最小二乘原理求得的。还有一个求解线性方程 $Ax+b=0$ 的指令是 linsolve，其基本格式如下：

```
[x0,kerA]=linsolve(A,b [,x0])
```

其中，A 为 $n\times m$ 矩阵；b 为 $n\times 1$ 的列向量；x0 为实数向量，作为方程求解的初值；kerA 为 $m\times n$ 矩阵（A * kerA=0）。下面给出一些实例，说明选择指令需要用户的试验，关键需要考察解的结果是否正确，这可以通过将解代回原方程检验其精度来实现。另外还要看计算速度如何。

例 7.6 电阻电路网络系统（满秩方程求解）

电阻电路网络系统如图 7.7 所示。根据欧姆定律和柯西霍夫定律，可以得到该网络的系统方程：

$$\begin{gathered}
i_1R_1+(i_1-i_2)R_2+(i_1-i_3)R_3+V_1=0\\
(i_2-i_1)R_2+(i_2-i_3)R_5-V_2=0\\
(i_3-i_1)R_3+i_3R_4+(i_3-i_2)R_5=0
\end{gathered}$$

上式可以用矩阵形式表达：

$$\begin{bmatrix} R_1+R_2-R_3 & -R_2 & -R_3 \\ -R_2 & R_2+R_5 & -R_5 \\ -R_3 & -R_5 & R_3+R_4+R_5 \end{bmatrix} \begin{Bmatrix} i_1 \\ i_2 \\ i_3 \end{Bmatrix} = \begin{Bmatrix} -V_1 \\ V_2 \\ 0 \end{Bmatrix}$$

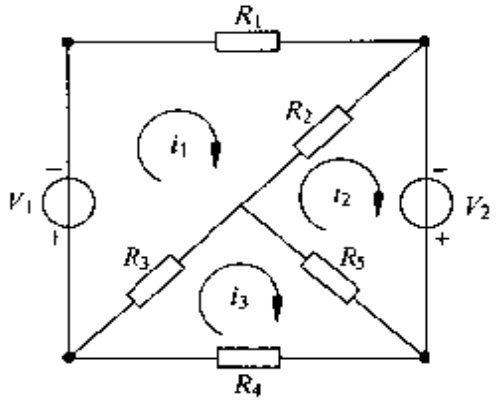


图 7.7 电阻电路网络系统

这是一个多变量求解的矩阵方程。在该系统应用中一般网络电源电压及各个电阻是给定的，需要求解各环路的电流分量。下面是已知的电路参数：

$$R_1 = 2\Omega, R_2 = 29\Omega, R_3 = 8\Omega, R_4 = 15\Omega,$$

$$R_5 = 4\Omega, V_1 = -10\text{V}, V_2 = 25\text{V}$$

应用 SCILAB 编写的计算程序如下：

```
R=[2 29 8 15 4], V=[-10 25 0]';          // 输入参数
A=[R(1)+R(2)+R(3) -R(2) -R(3); ...
  -R(2) R(2)+R(5) -R(5); ...
  -R(3) -R(5) R(3)+R(4)+R(5)];           // 构造矩阵 A
[I,kerA]=linsolve(A,-V);                 // 计算电流向量
if rank(A) == max(size(A)) then          // 如果矩阵 A 是满秩的，则可
                                         // 使用以下指令
   I1= inv(A) * V;                       // 计算结果应该与以上
                                         // 计算结果相同
end
```

这里同时使用了 linsolve 与 inv 指令进行计算。对于满秩矩阵两者的计算结果是相等的。但是应该注意，linsolve 指令对 Ax+b=0方程形式；inv 指令对 Ax=b 方程形式，并只能在满秩矩阵情况下使用。计算结果如下：

```
I=[1.5604027    2.2248322    .7919463]'
```

可以用 clean(A * I) 指令检验计算结果是否等于 V。其中，clean 指令是对近似为零的值取整零值。

例 7.7 运动方程参数估计(超定线性方程求解)

本例将进行超定线性方程求解。所谓超定的含义可以由下面的多项式拟合或回归问题来解释。大家知道在地球大气层范围内自由落体满足抛物线形式的运动方程：

$$y(t) = -\frac{1}{2}gt^2 + v_0 t + y(0)$$

其中，y 是物体与地面的高度；g 是重力加速度；v_0 是物体的初速度；而 $y(0)$ 是物体的初始高度。下面是对某一物体降落运动时进行的实测数据。

t (sec.)	1	3	5	9	15	24
y (m)	2995	2964	2884	2620	1931	225

根据运动方程及实测数据，可以写成以下的矩阵形式：

$$\begin{bmatrix} 1^2 & 1 & 1 \\ 3^2 & 2 & 1 \\ 5^2 & 5 & 1 \\ 9^2 & 9 & 1 \\ 15^2 & 15 & 1 \\ 24^2 & 24 & 1 \end{bmatrix} \begin{Bmatrix} -\frac{g}{2} \\ v_0 \\ y(0) \end{Bmatrix} = \begin{Bmatrix} 2995 \\ 2964 \\ 2884 \\ 2620 \\ 1931 \\ 225 \end{Bmatrix}$$

从上面的方程中可以看到共有 3 个需要估计的运动参数，而共有六组数据，大于代定参数个数，或者说，线性方程矩阵的行向量个

数大于列向量个数。因此这是一个典型的超定方程求解的数学问题。而在实际应用中它表现为数据回归、曲数拟合或参数估计等工程问题。下面是求解这个问题的程序：

```
t=[1, 3, 5, 9, 15, 24]';                    // 观测时刻
b=[2995 2964 2884 2620 1931 225]';          // 测量数据
z=b;A=[t.^2, t, ones(t)];                   // 构造矩阵
x=pinv(A) * b;                              // 参数估计计算
x1=A\b;                                     // 结果应该同以上计算结果
g=-2 * x(1); v0=x(2); z0=x(3);              // 输出估计值
xset("mark size",2); xsetech([0 0 1 1],[0,28,0,3000]);
plot2d(t,z,style=-1);
t1=linspace(0,24,25);plot2d(t1,-0.5 * g * t1.^2+v0 * t1+z0);
```

上面的指令 pinv 是矩阵的伪逆计算，这是进行求解超定方程的标准算法。即

$$\hat{x} = \text{pinv}(\boldsymbol{A}) = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}b$$

应用该方式计算的参数估计值满足最小估计方差：

$$\min \| \boldsymbol{A}\hat{x} - b \|_2$$

使用上面的程序，可得到以下计算结果和图形(见图 7.8)。

[g, v0, z0] = [9.830, 2.452, 2997.7]

例 7.8 基于符号的矩阵求逆计算

SCILAB 提供了可以与 MAPLE 软件(符号运算商用软件)连接进行符号运算的模式。所谓符号运算就是实现解析解的公式推导。我们知道数值解在应用中有很大的局限性，而解析解可以提供更全面、深入的理解现象，分析机理，解决问题的有用信息。在本例中介绍基于符号的矩阵求逆计算。这也是 SCILAB 本身(不借助于 MAPLE)可以完成基于符号运算的指令之一。例如计算以下矩阵的逆：

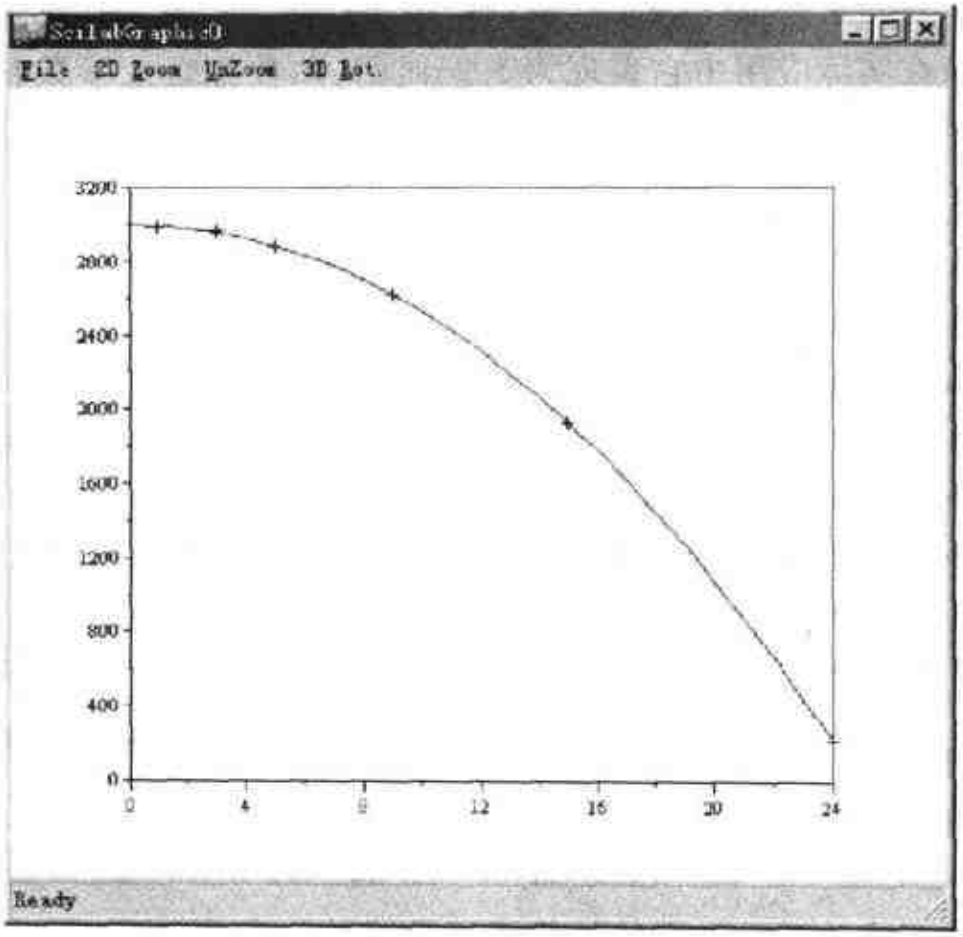


图 7.8 自由落体运动方程曲线

$$a=\begin{bmatrix} s & s^2 & \dfrac{-1}{1-s^2} \\ 1-s^2 & 1+s & 0 \\ -s & 0 & 0 \end{bmatrix}$$

```
s=poly(0,'s');a=[s,s^2,-1/(1-s^2); 1,1+s,0; -s, 0, 0];
// 构造矩阵
b=inv(a); clean(a * b);
// 求逆并验证
```

根据以上 SCILAB 语句，可以很方便地得到如下形式的结果：

$$b = \mathrm{inv}(a) = \begin{bmatrix} \frac{0}{1} & \frac{0}{1} & -\frac{1}{s} \\ \frac{0}{1} & \frac{-1}{-1-s} & \frac{-1}{-s-s^2} \\ \frac{-1+1.570\times10^{-16}s+s^2}{1} & \frac{s^2-s^3}{1} & \frac{-1+s}{1} \end{bmatrix}$$

可以通过验算 a * b 是否等于单位阵来验证计算结果是否正确。应用指令 clean(a * b)之后可以看到一个标准的单位阵结果。应该注意，目前的 SCILAB 指令 inv 在运算上有局限性：首先，它只适用于单变量符号操作；其次，矩阵各单元必须是有理多项式。

7.4 非线性方程组求解

下面介绍非线性方程求解。其基本格式如下：

[x [,v [,info]]]=fsolve(x0,fct [,fjac] [,tol])

其中，x0 是实数向量，作为方程求解的初值；fct 是用字符串表达的方程式；fjac 也是用字符串表达的方程式；tol 是实数值，为方程求解中止时的相对误差，其缺省值设为 tol=1.d-10；x 是方程过零点向量解；v 是实数向量，相当于方程 v(x)的解；info 输出使用该指令求解的效果信息(0：输入方式有问题；1：求解结果达到 tol 要求；2：求解结果未达到 tol 要求，但是程序以最大迭代次数中止计算；3：tol 给定值太小；4：计算结果不收敛)。下面给出几个实例。

例 7.9 一维非线性方程求解

现在计算两个非线性方程：

方程 1 $f(x) = e^{-x} - x = 0$

方程 2 $g(x) = x^2 + 4\sin(x) = 0$

下面是完成计算的指令：

```
deff('[y]=fsol1(x)','y=exp(-x)-x');          // 设定方程 1 函数
[x f]=fsolve(10,fsol1);                      // 求解方程 1
deff('[y]=fsol2(x)','y=x^2+4 * sin(x)');     // 设定方程 2 函数
[x1 g1]=fsolve(-10,fsol2);                   // 求解方程 2(解 1)
[x2 g2]=fsolve(10,fsol2);                    // 求解方程 2(解 2)
```

经过计算可以得到方程 1 的单值解与方程 2 的双值解，分别为[x, x1, x2] = [.5671433 0. 1.9337538]。需要注意的是，该 fsolve 指令是应用了 Powell 杂交算法。对于多解方程，必须通过选择不同初始值来得到结果，例如上面的方程 2 求解。当不知道解的个数时，可以借鉴图形方式(见图 7.9)。

```
a=-3;b=3;x=[a : 0.01 : b]'; z=0 * x;
// 应用 a,b 来控制方程求解及显示范围
y1=exp(-x)-x;          // 方程 1: x=0.5671
y2=x.^2-4 * sin(x);    // 方程 2: x1=0, x2=1.9338
xsetech([0,0,1.0,0.5]);plot2d(x,[z y1]);xtitle('y=exp(-x)-x');
xsetech([0,.5,1,.5]);plot2d(x,[z y2]);xtitle('y=x.^2-4 * sin(x)');
```

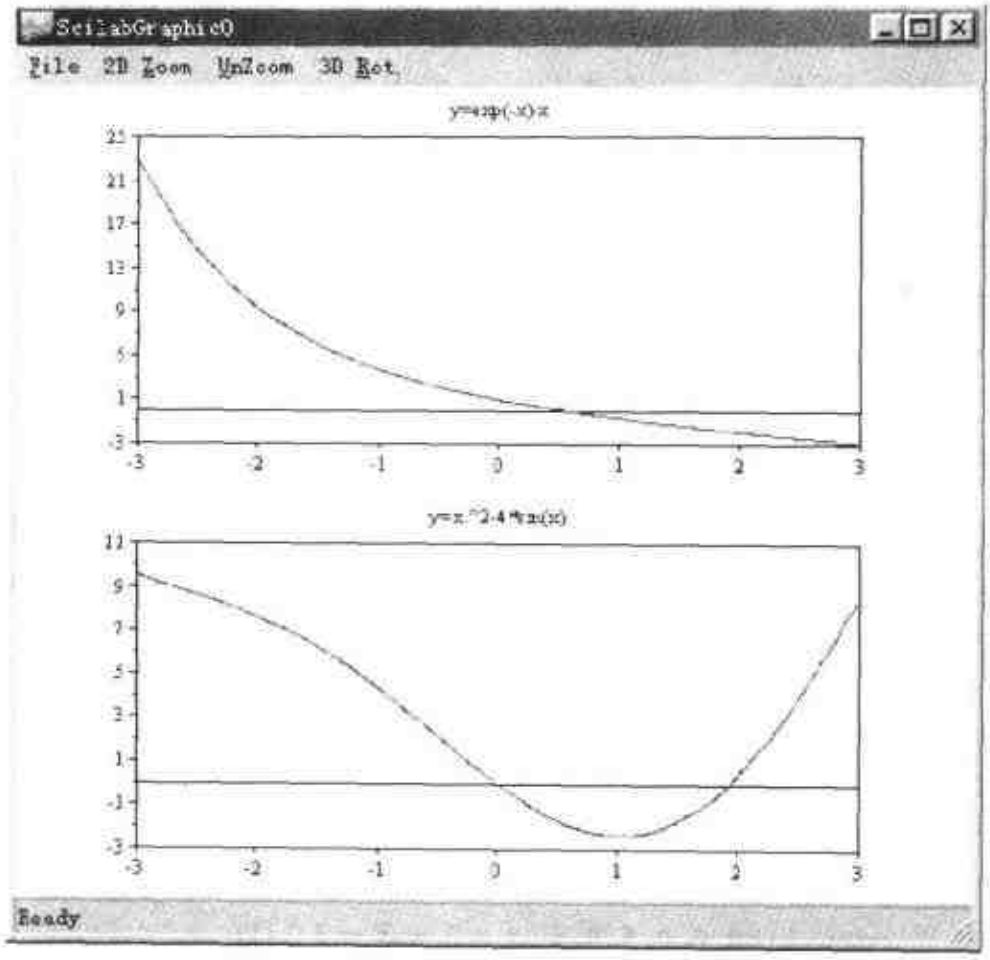
例 7.10 二维非线性方程求解

下面对一个二维非线性方程进行求解：

$$f(x) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} x_1^2 \\ x_2^2 \end{Bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{Bmatrix} c \\ c \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

可以看出这是二次抛物线方程。其中 c 是代定参数。我们希望考察这个方程的可能解的个数。这一点可以通过作图看出。当 c 分别等于 1.0, 0.25, −0.5, −1.25 时，该方程共有 0,1,2,4 四个解。

```
a=[1,0;0,1];b=[0,-1;-1,0];          // 设定各矩阵元素为固
                                    // 定值
```



上图：$f(x)=e^{-x}-x=0$　下图：$g(x)=x^2+4\sin(x)=0$

图 7.9　一维非线性方程图形解

```
c=[0.25;0.25];                                    // 设定各矩阵元素为固
                                                  // 定值
deff('[y]=fsol(x)','y=a*x^2+b*x+c');              // 设定方程函数
[x1 f1 IN1]=fsolve([100;100],fsol);               // 求解方程(可能解 1)
[x2 f2 IN2]=fsolve([10;-100],fsol);               // 求解方程(可能解 2)
[x3 f3 In3]=fsolve([-100;100],fsol);              // 求解方程(可能解 3)
[x4 f4 In4]=fsolve([-100;-100],fsol);             // 求解方程(可能解 4)
```

以上，设定了 4 组不同初始值计算，其中应该观察 xi，fi，与 INi 的输出值，以判别无解、有解和重复解。下面是本例的计

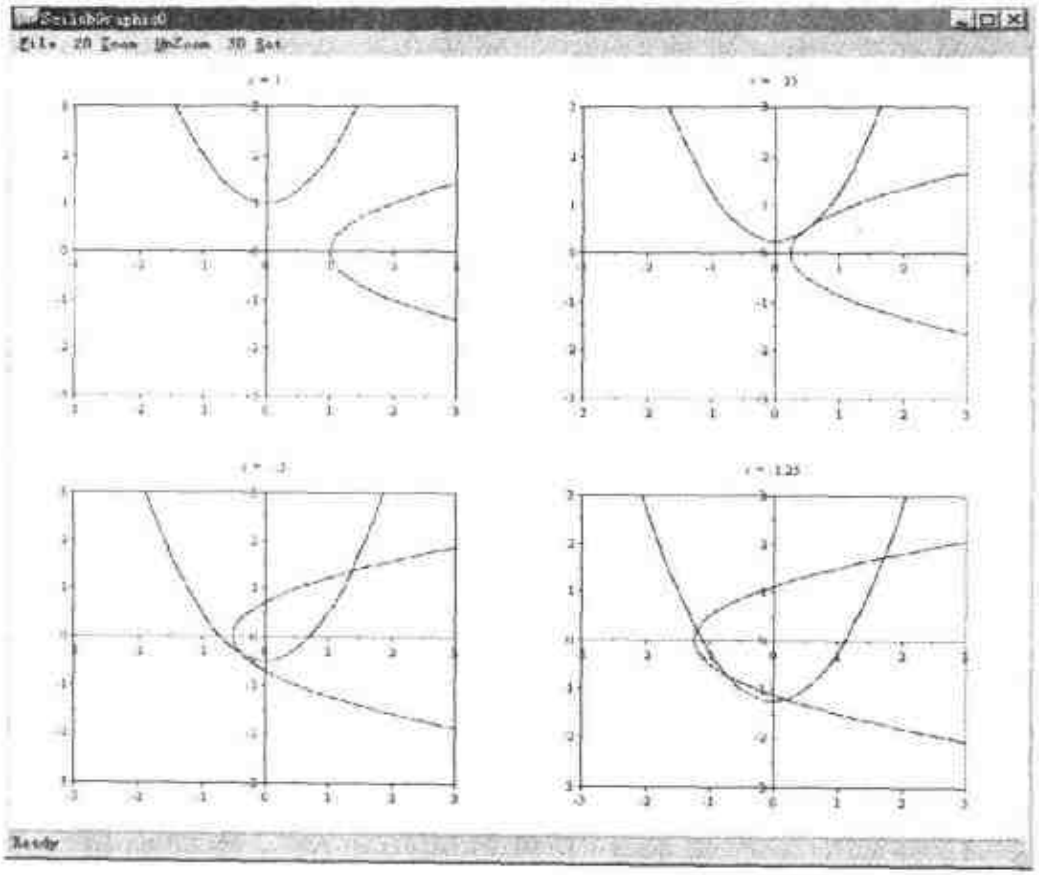
算结果：

c＝1.0 无解； 全部 INi ＝ 4，且 fi ＞ 0.5。
c ＝ 0.25 一组解： x1 ＝ [0.5，0.5]，全部 INi ＝ 1，且 fi ≈ 0。
c ＝－0.25 两组解：x1 ＝ [1.366，1.366]，x2 ＝ [－0.366，－0.366]。
c ＝＋1.0 四组解： x1 ＝ [1.725，1.725]，x2 ＝ [0.207，－1.207]，
x3 ＝ [－1.207，0.207]，x4 ＝ [－0.725，－0.725]。

应用以下程序，可以用图形观察上述4种情况，理解非线性方程解的几何意义（见图7.10）。实际上，方程解的个数是由两条抛物线的交点个数表现的。

```
b=3;c=1; dc=0.75;
for i=1 : 4,
    x1=[c : 0.01 : b]';x2=sqrt(x1-c);
    if i==1 then
      xsetech([0,0,0.5,0.5],[-3 -3 -3 3]);
    end
    if i==2 then
      xsetech([0.5,0.0,0.5,0.5],[-3 -3 -3 3]);
    end
    if i==3
      xsetech([0.0,0.5,0.5,0.5],[-3 -3 -3 3]);
    end
    if i==4
      xsetech([0.5,0.5,0.5,0.5],[-3 -3 -3 3]);
    end,
    plot2d(x1,x2);plot2d(x1,-x2);
    plot2d(x2,x1);plot2d(-x2,x1,1,"084",);
```

```
    caption = "c = " + string(c); xtitle(caption);
    c=c-dc;
end
```



左上图:c=1.0　无解;右上图:c=0.25　一组解;
左下图:c=-0.5　两组解;右下图:c=-1.25　四组解

图 7.10　二维非线性方程图形解

7.5 线性规划

线性规划是求解线性优化问题的通常算法,该类问题的基本表达形式为

$$\text{Minimize } f(\boldsymbol{x})=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$$
$$\text{Subject to } \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$$
$$\boldsymbol{G}\boldsymbol{x}\leqslant\boldsymbol{h}$$
$$\boldsymbol{x}_{\mathrm{L}}\leqslant\boldsymbol{x}\leqslant\boldsymbol{x}_{\mathrm{U}}$$

其中，$\boldsymbol{x}$ 是变量，为 n 维列向量，它的优化解范围通常受到等式，不等式以及上下界范围3类形式的约束。线性规划问题规定它的目标函数方程，约束方式均是线性形式。因此其他的矩阵或向量应该是常数形式。$\boldsymbol{c}$ 同样是一个 n 维列向量；$\boldsymbol{A}$ 是 $me\times n$ 维矩阵；$\boldsymbol{G}$ 是 $md\times q$ 维矩阵；$\boldsymbol{b}$ 是 me 维列向量；$\boldsymbol{h}$ 是 md 维列向量；$\boldsymbol{x}_{\mathrm{L}}$ 与 $\boldsymbol{x}_{\mathrm{U}}$ 分别为 $\boldsymbol{x}$ 变量的上下界范围约束的 n 维列向量。完成线性规划求解的 SCILAB 基本指令调用格式如下：

```
[xopt,lagr,fopt]=linpro(c,C,d [,x0])
[xopt,lagr,fopt]=linpro(c,C,d,xL,xU [,x0])
[xopt,lagr,fopt]=linpro(c,C,d, xL,xU,me [,x0])
[xopt,lagr,fopt]=linpro(c,C,d, xL,xU,me,x0 [,imp])
```

其中，c 为 n 维的实数列向量；C 是 $(me+md)\times n$ 维矩阵，无约束时设定 C = []；d 是 $me+md$ 维列向量，无约束时设定 d = []；xL 与 xU 分别为 x 变量的上下界范围约束的 n 维列向量；me 是等式约束方程的个数；x0 是优化计算的初始计算值；imp 是选择项，整数值(被建议设定为 imp = 7,8,…)；xopt 与 fopt 分别为优化解的变量值向量与优化值结果；lagr 是拉格朗日(Lagrange)乘子向量。在实际计算目标函数最优解时，约束条件是按照 Lagrange 因子 λ_i 与约束条件函数 $\varphi_i(x)$ 乘积后按照下面新的目标函数进行优化计算：

$$\text{Minimize } \Phi(x)=f(x)+\Sigma\lambda_i\varphi_i(x)$$

因此 Lagrange 乘子向量 lagr 的长度一般为 $n+me+md$。前 n 个元素对应上下界范围(或边界)约束的 Lagrange 乘子值，中间 me 个元素对应等式约束乘子值，后 md 个元素对应不等式约束乘子

值。如果任何元素值输出均为零，则表明该约束未起作用。对于边界约束，当 Lagrange 乘子输出为负值时，表明下边界起作用；为正值时，表明上边界起作用。

例 7.11 三维线性规划求解

下面是三维线性规划问题：

$$\begin{aligned}
\text{Minimize} \quad & f(x) = 3x_1 - 5x_2 - 2x_3 \\
\text{Subject to} \quad & x_1 + 3x_2 = 5 \\
& x_1 + x_2 - x_3 = 2 \\
& 2x_1 - x_2 \leqslant 3 \\
& x_1 + x_2 - x_3 \leqslant 25 \\
& 0 \leqslant x_1 \leqslant 5 \\
& 0 \leqslant x_2 \leqslant 10 \\
& 0 \leqslant x_3 \leqslant 3
\end{aligned}$$

对于该问题，方程形式为

$$c = \begin{Bmatrix} 3 \\ 5 \\ -2 \end{Bmatrix}, x = \begin{Bmatrix} 3 \\ 5 \\ -2 \end{Bmatrix}, A = \begin{bmatrix} 1 & 3 & 0 \\ 1 & 1 & -1 \end{bmatrix}, b = \begin{Bmatrix} 5 \\ 2 \end{Bmatrix},$$

$$G = \begin{bmatrix} 1 & 3 & 0 \\ 1 & 1 & -1 \end{bmatrix}, h = \begin{Bmatrix} 5 \\ 2 \end{Bmatrix}$$

$$xL = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}, xU = \begin{Bmatrix} 5 \\ 10 \\ 3 \end{Bmatrix}$$

下面是有关的计算程序：

```
c = [3;5;-2]; A=[1,3,0;1,1,-1]; b=[5;2]; G=[1,3,0;1,1,
-1]; h=[5;2];
xL=[0;0;0]; xU=[5;10;3]; me=2;
C=[A;G]; d=[b;h];
```

```
[xopt,lagr,fopt]=linpro(c,C,d, xL,xU,me)
```

运行程序后,计算结果如下:

```
xopt = [ 0.5, 1.5, 0], lagr = [0, 0, 0, -1, -2, 0, 0], fopt = 9.
```

将 xopt 值代入目标函数方程,可以验证其解 fopt = 9 是正确的。Lagr 值表明只有两个等约束方程起作用。

Scicos——图形化动态模型仿真器

8.1 Scicos基本内容介绍

Scicos（SCILAB connected object simulator）是一种基于SCILAB平台的图形化动态模型仿真器，在功能上类似于MATLAB中的SIMULINK。该仿真器用图形模块的方式来直观、有效地表现物理过程或数字模型中的数据传递、演算、变换、显示等过程。Scicos本身不独立给出版本定制，而是以SCILAB的版本为准，它可以理解为是SCILAB内建的工具箱。下面是Scicos的基本特征：

（1）为基于用户图形界面(GUI)编辑器的动态建模方法；

（2）可以采用超级模块方式实现多极模块结构；

（3）适于各种应用的标准模块库；

（4）可以应用C或FORTRAN语言的动态连接方式，或SCILAB语句方式重建用户定义的专用模块；

（5）采用强有力的、包容各种模型（如连续、离散以至事件的单模型或混杂模型）的统一化处理方式；

（6）快速的仿真计算过程。

Scicos 可以用几种方式开始运行。在此，先介绍一种最常用的方式来建立 Scicos 的运行方式。在 SCILAB 提示符下，可以通过输入以下命令启动 Scicos：

```
-->scicos();
```

这时 SCILAB 将自动弹出一个窗口（见图 8.1），窗口名称为“Untitled”，将它称为“Scicos 图形化动态模型仿真器主窗口”。它也是完成模型设计的编辑器平台。

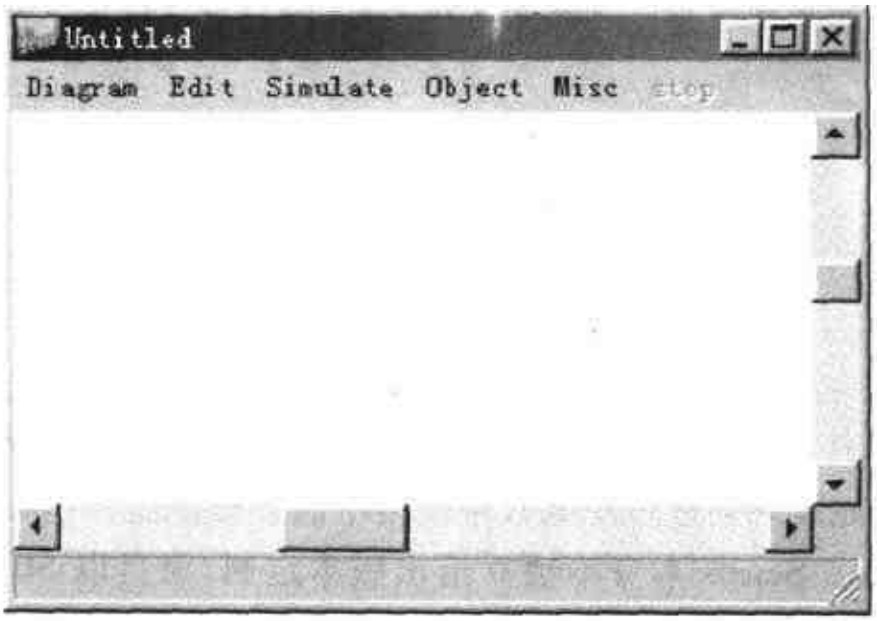


图 8.1 Scicos 图形化动态模型仿真器主窗口

在窗口上方包括 5 个下拉菜单。它们的含义分别如下：

【Diagram】	【图形】
Replot	模块图形刷新
New	创立新文件
Region to Supper Block	将区域内模块生成超级模块
Purge	将超级模块分解为原模块
Rename	更改文件名

Save	存文件
Save as	另存文件
Load	读文件
Load as Palette	读用户定义的子模块库
Save as Palette	存用户定义的子模块库
Save as Interf Func	存用户定义函数模块文件
Set Diagram Info	用户信息记录
Navigator	超级模块结构浏览
Export	输出PS文件或图形
Export all	输出全部为PS的文件或图形
Quit	退出Scicos
【Edit】	【编辑】
Palettes	模块
Context	模块中的变量设定
Smart_Move	模块的特殊移动(模块间连线保持为水平或竖直)
Move	模块移动(模块间连线可能为斜线)
Copy	模块复制
Copy Region	区域内模块复制
Replace	替换(先点击替换模块,后点击被替换模块)
Align	模排列对齐
Link	模块连接
Delete	删除
Delete Region	区域内模块删除
Add new block	添加用户定义函数模块文件
Flip	模块水平方向翻转

Undo	退回上次操作
Pat editor	模块编辑器
【Simulate】	【仿真】
Setup	仿真参数设置
Compile	编译
Eval	计算当前变量值
Run	启动仿真
【Object】	【对象】
Open/Set	打开/设置
Resize	重置尺寸
Icon	图标
Icon Editor	图标编辑
Color	颜色
Label	标记
Get Info	读/写信息
Identification	标识
Documentation	文件
【Misc】	【杂项】
Window	窗口
Background color	背景颜色
Default link colors	缺省连线颜色
ID fonts	标识字型
Aspect	模块显示形式设置
Add color	用户特定颜色设置
Short cuts	快速键操作(SCILAB 主窗口显示快速键操作内容，如键入“m”键后，可以选移动模块)。
Zoom in	显示放大

Zoom out	显示缩小
Help	帮助(点击对象或菜单内容获得帮助)
Calc	计算(转入 SCILAB 主窗口进行计算,键入"return"后退回本 Scicos 窗口)
【Stop】	【停止仿真】

8.2 Scicos 子模块库介绍

Scicos 内建了一些标准模块库。每个模块是按照功能类别分属在不同的子模块库中。调用这些模块,可以在【Edit】对话键中选择"Palette"。由此自动弹出一个有选择内容的窗口(见图 8.2)。

图 8.2 选择窗口

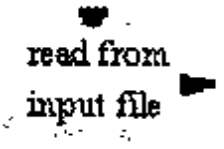
该选择内容窗口共有7个选择项，分别对应于7个子模块库。这些子模块库提供的每一个图形式模块是Scicos的标准模块。每一个模块相当于流程图中的模块，它可以完成一个或若干个特定的功能。下面对7个子模块库中的每一个标准模块进行基本注释：

【Inputs_Outputs】 输入输出子模块库

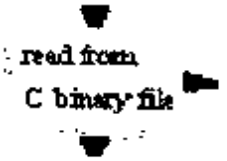
超级模块事件信号输出端口

超级模块事件信号输入端口

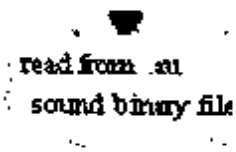
事件信号发生器



读输入文件



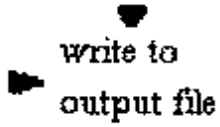
读C文件



读AU文件

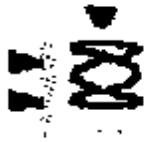
写 C 文件

写 AU 文件



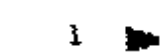
写输出文件

数字显示端口



双路信号相平面图(SCOPXY_f 模块方式)

双路信号相平面图(ANIMXY_f 模块方式)



常数值,或常数向量发生器

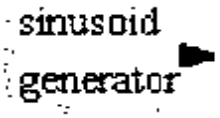
垃圾箱模块,用于接收一些可忽略的数据

基于时间的表格函数模块

事件信号示波器

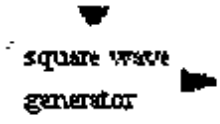
时间发生器

数字信号示波器



正弦信号发生器

锯齿波信号发生器



方波信号发生器

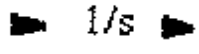
多路数字信号示波器

多路事件信号示波器

数字信号输入端口

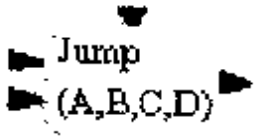
数字信号输出端口

【Linear】 线性单元子模块库



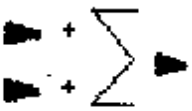
积分模块（连续信号）

单位延迟模块(离散信号)

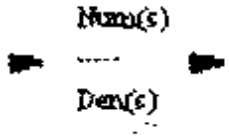


连续型、可跳跃式线性状态空间系统模块

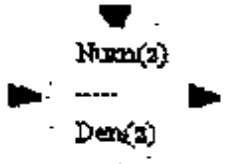
零阶保持采样单元



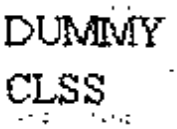
多通道数字信号加和模块（通道可设定负值加和）



单输入单输出线性连续系统传递函数

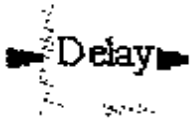


单输入单输出线性离散系统传递函数



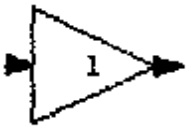
哑连续系统模块

移位寄存器

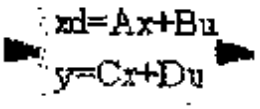


延时器

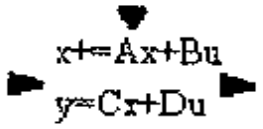
时变延迟模块



增益器



连续型线性系统模块



离散型线性系统模块

多通道数字信号加和模块(各通道可以是不等频率事件)

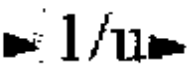
【Non_Linear】 非线性单元子模块库

离散单元模块

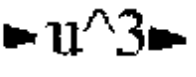
饱和单元模块

基子图形查表模块

$y(i) = a^{u(i)}$ 函数计算模块

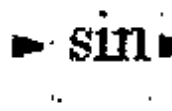


$y(i) = 1/u$ 函数计算模块

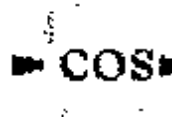


$y(i) = u^b$ 函数计算模块

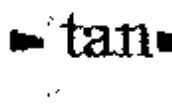
多通道数字信号乘积模块



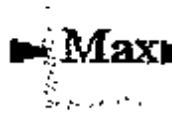
正弦单元模块



余弦单元模块

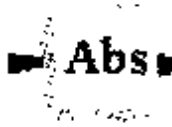


正切单元模块



取大单元模块

取小单元模块

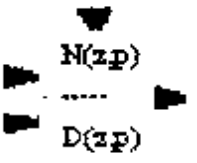


绝对值单元模块

对数单元模块

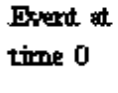
单通道线性插值单元模块

双通道线性插值单元模块



离散型线性自适应系统模块

【Event】 事件子模块库



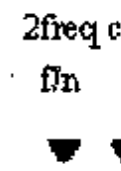
事件启动触发模块

事件时钟分频模块

事件延迟器模块

逻辑和模块

事件逻辑和

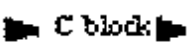


事件双时钟频率，低分频(f/n)模块

Trash	垃圾箱模块，用于接收一些可忽略的数据
【Treshold】	阀值子模块库
+ to -	“正-负”触发器（当数字信号由正变负时产生事件触发信号）
- to +	“负-正”触发器（当数字信号由负变正时产生事件触发信号）
Zcross	“过零”触发器（当数字信号通过零点时产生事件触发信号）
GENERAL	通用多通道“过零”触发器（当数字信号通过零点时产生事件触发信号）
【Others】	其他子模块库
Scifunc	交互式定义模块
Text	文字注释模块
Mem	存储器模块

通用过零点监测模块

FORTRAN 语言函数模块



C 语言函数模块

超级模块

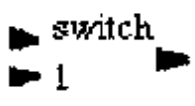
【Branching】 分支子模块库

（未注明）

多通道（通道个数可设）数字信号合成器
（单通道输出信号成为多维向量信号）

分路器（多维向量输入信号降为多路低维信号输出）

多路继电器

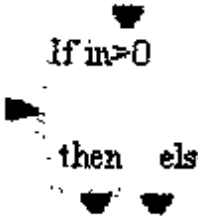


多输入(数字信号通道个数可设置)单输出选择器

多输入(数字与事件通道个数可设置)单输出选择器

单输入多输出(数字与事件通道个数可设置)选择器

事件选择器



基于条件的事件选择器

事件加和单元(各通道可以是不等事件频率)

8.3 应用Scicos子模块库的一般注意事项

本节对应用 Scicos 子模块库时的注意事项做一介绍。

(1) Scicos 在仿真处理中有两种信号方式:一种是常规的数字信号;另一种称为事件信号。事件信号又称为作用信号,主要

用于控制各模块进行数字信号的计算、推演或显示的时刻。模块中的红色图示部分对应事件信号；黑色图示部分对应数字信号。

(2) 因此在 Scicos 中标准模块可以分为两大类：事件模块与数字模块。事件模块用全红颜色标注。由于本书中无法表现红色，在上节的图示标准模块中，该类模块是以较浅的灰度显示的。其他非全红(或非全浅色灰度)显示的模块是数字模块。

(3) Scicos 的模型设计应该至少包括一个事件模块，目的是用于整个模型仿真时的同步作用信号。

(4) 当 Scicos 的模型设计中包括多个事件模块时，事件信号之间一般是非同步的，因此应该尽量避免使用多个事件模块。

(5) 全部标准模块的上下端口分别对应于事件信号的输入与输出。而模块的左右端口分别对应数字信号的输入与输出(当然，如果对该模块进行“模块水平方向翻转”操作后，则左端口可以对应输出，右端口可以对应输入)。

(6) 全部 Scicos 标准数字信号模块可以分为两大类：当标准模块上或下方有红色端口时，称它为有事件信号端口模块；否则，称为无事件信号端口模块。

(7) 在 Scicos 模型中，无事件信号端口模块是应用事件内传递(inherit)功能实现仿真同步的，例如增益器模块。所谓“inherit”意为该模块是由数字信号通道传递事件信号的。

(8) 对于有事件信号端口模块，一般是有关于事件内传递(inherit)功能的设定。它的缺省设定是“inherit”＝0，表明该模块必须通过外部端口来传递事件信号；当设定改为“inherit”＝1 时，该模块的事件信号端口将自动在模块图形

显示中消失。此时，该模块转换为如同无事件信号端口模块方式运行。

(9) Scicos 中的多数标准模块是有设定值对话窗口。点击该类模块，它将自动弹出一个对话窗口。Scicos 已经给出每个设定值的缺省值，用户只需要对不同的缺省值进行更改即可。注意，"[]"代表"空值"设定。

(10) 对于多端口模块的端口排列次序如下：事件端口是从左到右升序排列，即最左边端口是第一端口；数字端口是从上到下升序排列，即最上边端口是第一端口。

(11) 端口设定可以包含事件的、数字的或者是输入、输出的。进行设定时对话窗口对其形式是有说明的。下面只介绍设定的含义。例如设定输出"port sizes"=[2, 8, 1]，意味着有 3 个输出向量端口，第一端口为 2 值向量，第二端口为 8 值向量，最后的端口为单值向量。该模块最后将自动显示 3 个输出向量端口。

8.4 Scicos 中的 3 类基本模块

下面对 Scicos 归纳的 3 种类型基本模块分别加以介绍。

1. 常规模块(regular basic block, RBB 模块)

Scicos 中的大多数模块属于该类模块。该模块的基本数学表达形式为

$$\dot{x} = f(t, x, z, u, p, n_e) \tag{8.1}$$

其中，t 是时间变量；x 是连续状态变量；z 是离散状态变量；u 是输入变量；p 是常数向量；f 是向量形式的非线性函数；n_e 是事件输入码态(activation code)总数。如果事件输入端口记为 i_1，

i_2, …, i_n,则

$$n_e = \sum_{j=1}^{n} 2^{i_j-1} \tag{8.2}$$

公式(8.1)是按照微分形式表达的连续状态变量 x 的变化。实际的状态变化是按下式在作用信号触发下,阶跃式地完成计算:

$$x(t_e) = g_c(t_e, x(t_e^-), z(t_e), u(t_e), \boldsymbol{p}, n_e) \tag{8.3}$$

$$z(t_e) = g_d(t_e, x(t_e^-), z(t_e^-), u(t_e), \boldsymbol{p}, n_e) \tag{8.4}$$

其中 t_e 是事件时刻;g_c 与 g_d 分别是连续与离散函数。在事件信号作用下,该类模块的输出形式如下:

$$y(t) = h(t, x(t^-), z(t^-), u(t), \boldsymbol{p}, n_e) \tag{8.5}$$

在两个事件信号之间,上式将保持常数值。同时,该模块还计算内部传递的事件信号:

$$t_{evo} = k(t_e, z(t_e), u(t_e), \boldsymbol{p}, n_e) \tag{8.6}$$

这里 t_{evo} 是事件信号向量,其中每一个原素对应一个事件信号输出端口。

2. 过零点模块(zero crossing basic block,ZBB 模块)

该模块是当输入数字信号跨越零线时,或者说输入数字信号改变符号时,输出一个事件触发信号。该事件信号的幅值为单位1。这种类型符号主要是在门槛值子模块库中应用。

3. 同步模块(synchro basic block,SBB 模块)

同步模块是指该模块能够生成与输入事件信号同步的作用信号。该类模块只有一个事件信号输入端口,但是却有几个事件输出端口。通过数字信号的控制,该模块将事件输入信号直接转接到特定的事件输出端口。典型的该类模块包括"事件选择器"与"基于条件的事件选择器"。注意,常规模块的事件输入信号与其自生成的传递事件输出信号是不同步的。同样,过零点模块生成的事件信号滞后于真正过零点的时刻。

8.5 Scicos 应用实例

例 8.1 正弦波信号生成、放大与显示

现在，开始第一个 Scicos 仿真实验。下面是一些基本步骤：

(1) 在 SCILAB 窗口下，打开 Scicos

```
-->scicos();
```

如此生成一个“Untitled”窗口(见图 8.1)。

(2) 在【Edit】菜单中，选择“Palette”，打开【Input_Ouput】子模块库。分别选择“Sinusoid Generator”(正弦发生器)、“Scope”(指示器)和“Event Generator”(事件发生器)。选择方法为单击被选目标，并将其拖至“Untitled”窗口，窗口内便包含该模块。

(3) 打开【Linear】子模块库。选择“Gain”(增益器)。以上各模块位置如图 8.3 所示。

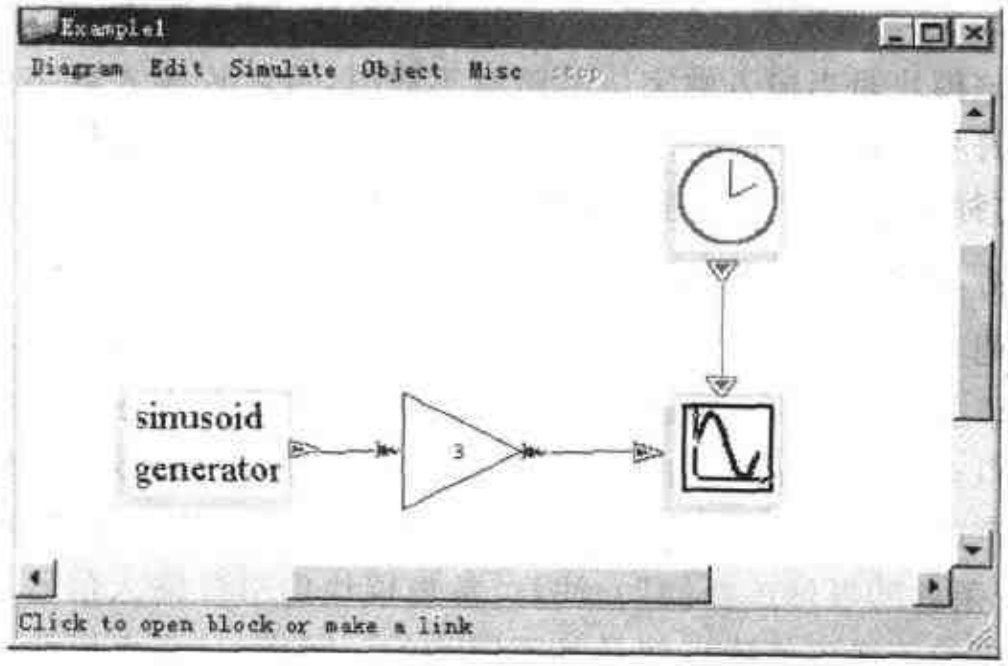


图 8.3 例 8.1 的 Scicos 模型

(4) 在【Edit】菜单中，选择“Link”。分别单击被连接的两个目标模块，这样完成一次连接。然后按图 8.3 的方式将全部模块连接。

(5) 在“Untitled”窗口内，单击“Gain”(增益器)，弹出一个对话窗口。按图 8.4 中的数值改变“Amplitude”(幅值)与“Frequency”(频率)。单击“Finish”(完成)结束对话窗口。

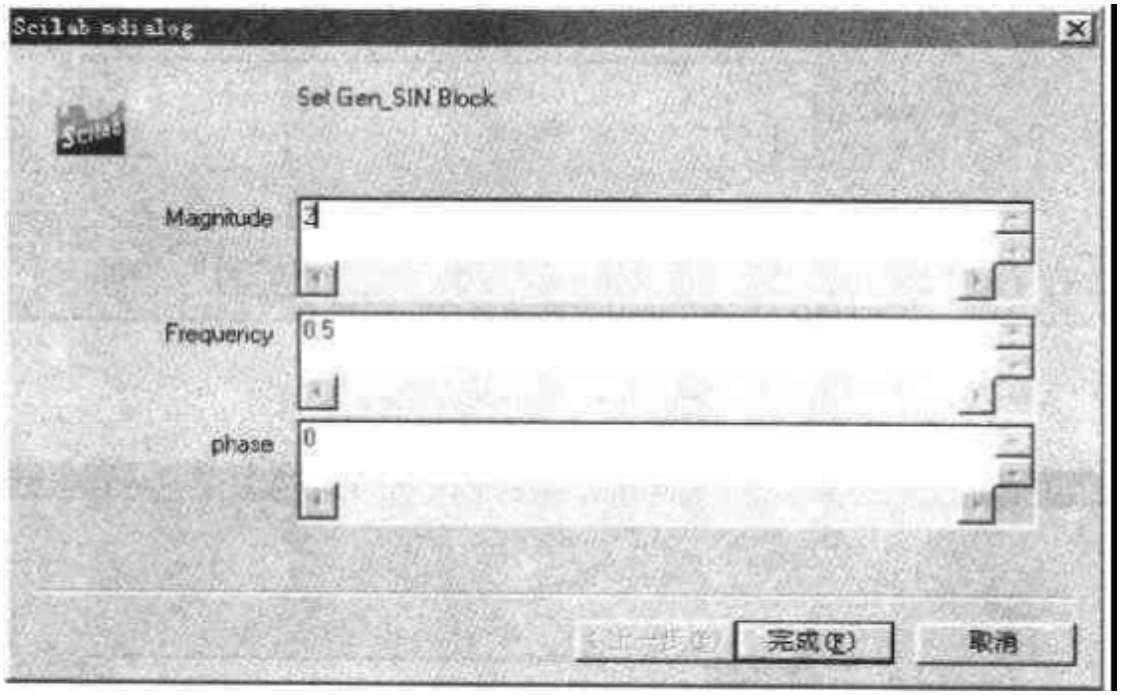


图 8.4 正弦波参数设置窗口

(6) 用以上方式，将“Gain”(增益器)设定值改变为“3”。

(7) 单击“Scope”(指示器)，改变其中“Ymin”为“－10”，“Ymax”为“10”。

(8) 在【Simulate】菜单中，选择“Setup”来改变 Scicos 仿真过程中的参数。分别按图 8.5、图 8.6 设定参数。

(9) 在【Diagram】菜单中，选择“Save as”，将新文件存储在用户特定的子目录下。文件名为“Example1.cos”。

(10) 在【Simulate】菜单中，选择“Run”来启动仿真过程。

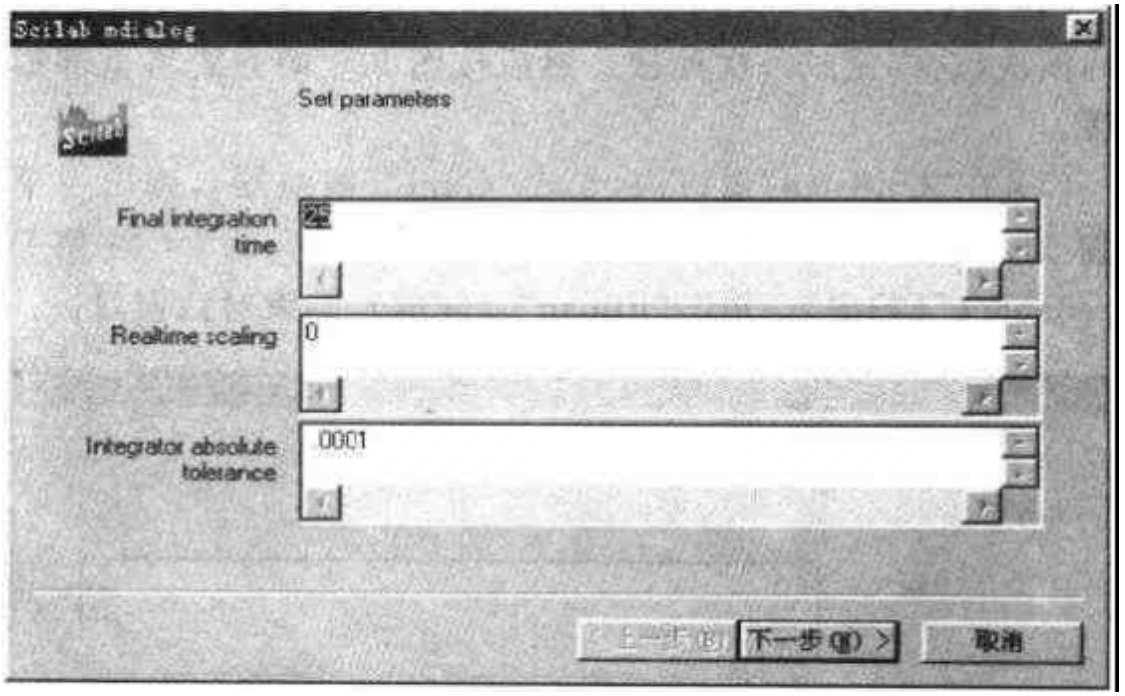


图 8.5 数字信号示波器设置窗口 1

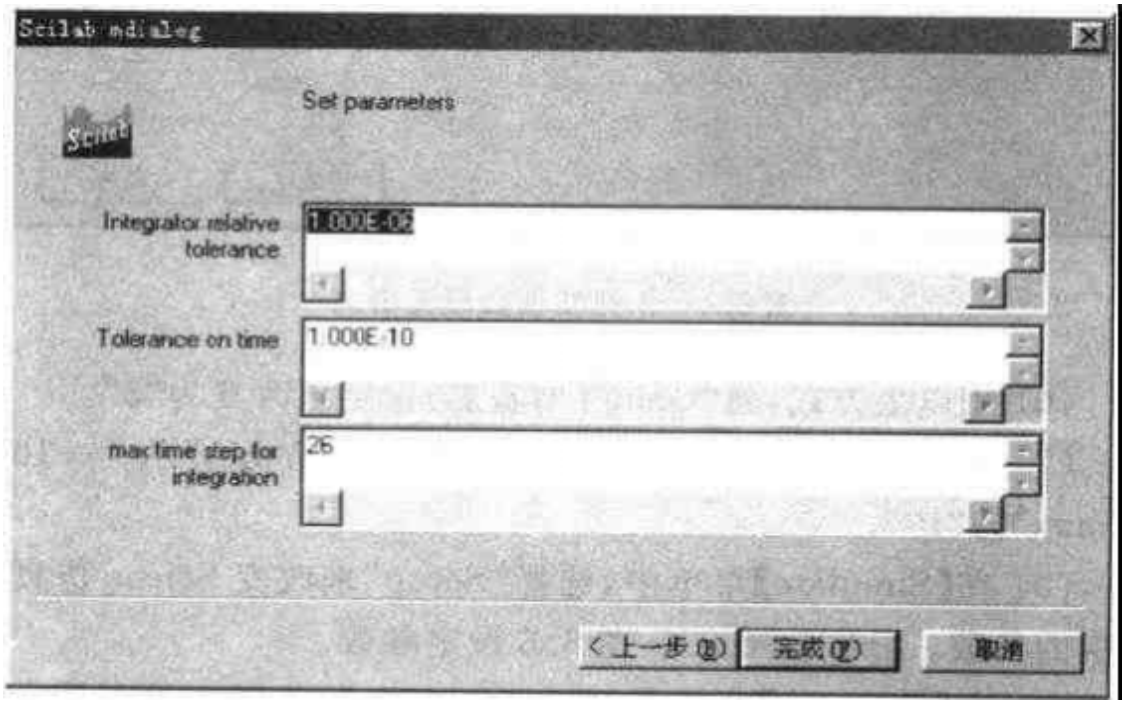


图 8.6 数字信号示波器设置窗口 2

Scicos 自动弹出显示窗口(见图 8.7)。

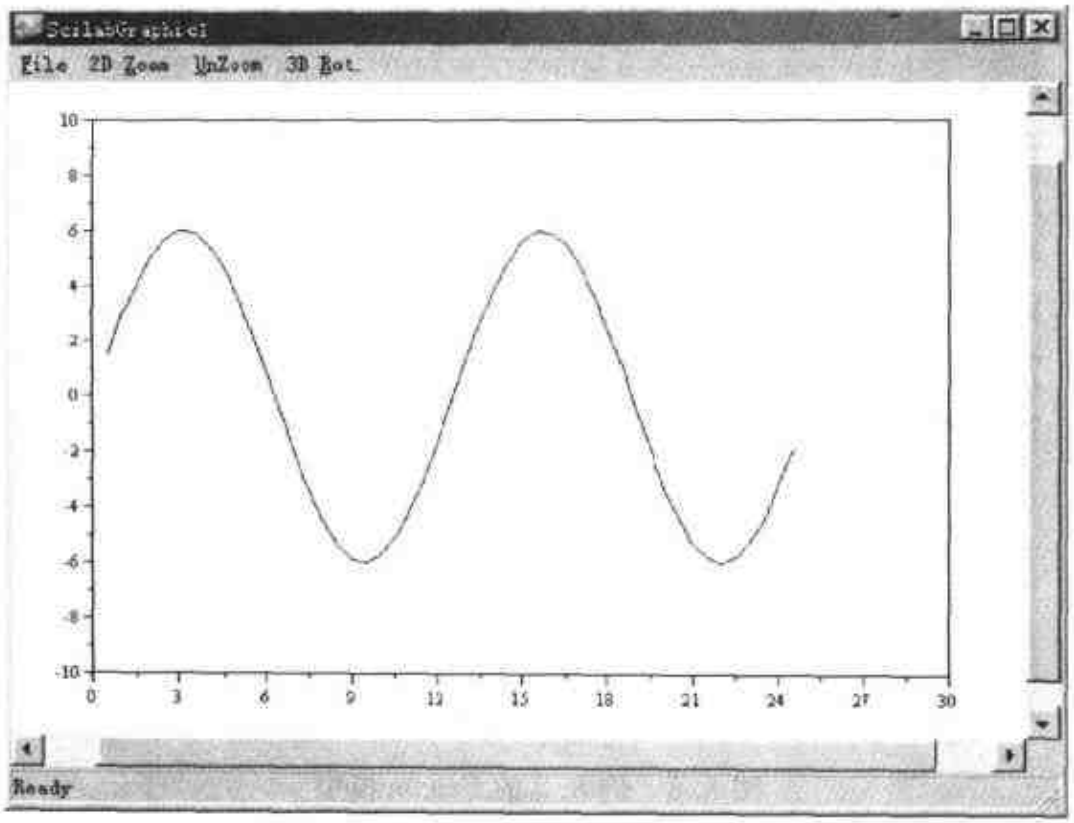


图 8.7 数字信号示波器显示仿真结果

可以看出,该仿真例题是一条正弦波信号,其频率为 0.5Hz,幅值为 6。该幅值分别为原正弦波幅值与增益器放大因子的乘积值。该波形显示的时间长度范围是 1～26。

例 8.2 超级模块的应用

Scicos 是应用图形化模块构造动态模型的。在这个例题中,将讲解超级模块的应用。图 8.8 显示了一个初始的如同例题 8.1 一样的动态模型。其中有两个通道的数字信号被显示。"MScope"是一个多通道数字信号示波器,它上面的一个输入通道是对正弦波的直接显示,而下面一个输入通道是对正弦波信号经过积分、放大处理后的显示。

注意,上图是 SCILAB 在 Linux 版本上生成的例图。它的界面形

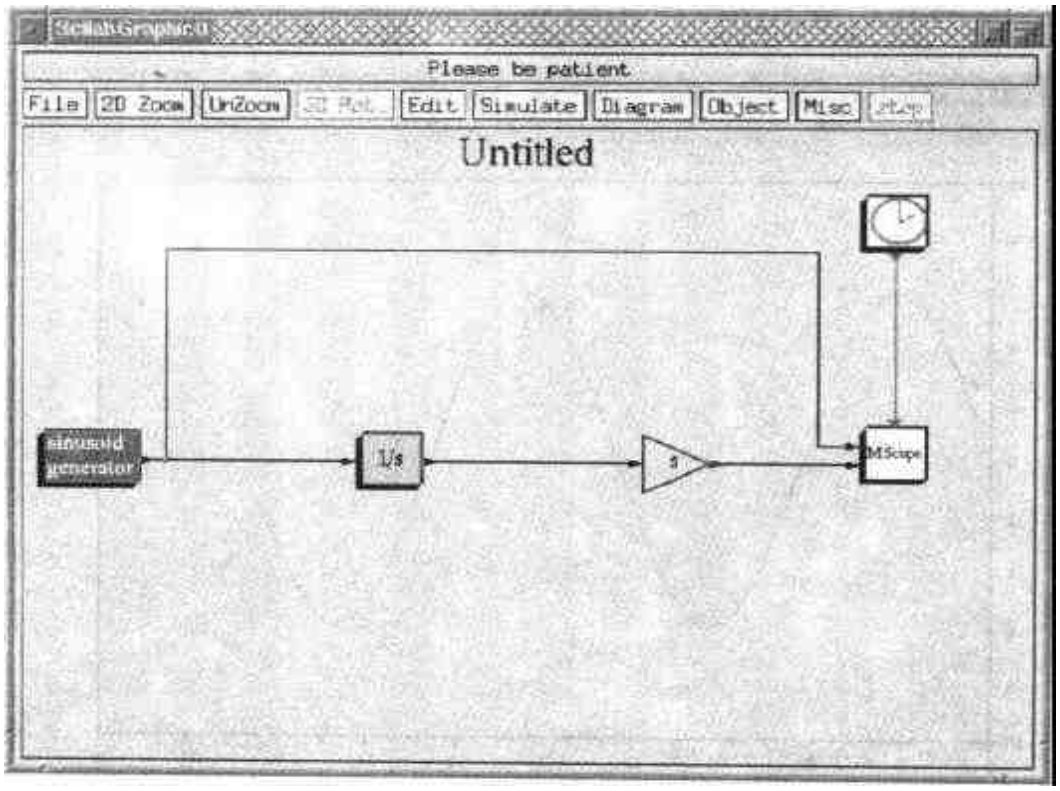


图 8.8 例 8.2 的 Scicos 模型

式与前面介绍的在 Windows 版本上的形式有所不同。但是在构造动态模型方法的基本步骤时，两者是基本相同的。此例题的构造步骤可以参考上一例题。需要提及的是，在 Scicos 模块之间连线时，如果从一根连线上另外引出连线(例如本例中在正弦波发生器与积分器连线之间的连线)，则该连线将传递相同的数字或事件信息到另一个模块输入端口(而不能再连接到信息连线上)。

下面将积分器与增益器两者集成为一个所谓“超级模块”。下面是基本步骤：

① 在图 8.8 的下拉菜单【Diagram】中，选择“Region to Supper Block”操作。

② 单击选定区域的两个对角点，在该区域范围内应该包括积分器与增益器两个模块，如图 8.9 所示。

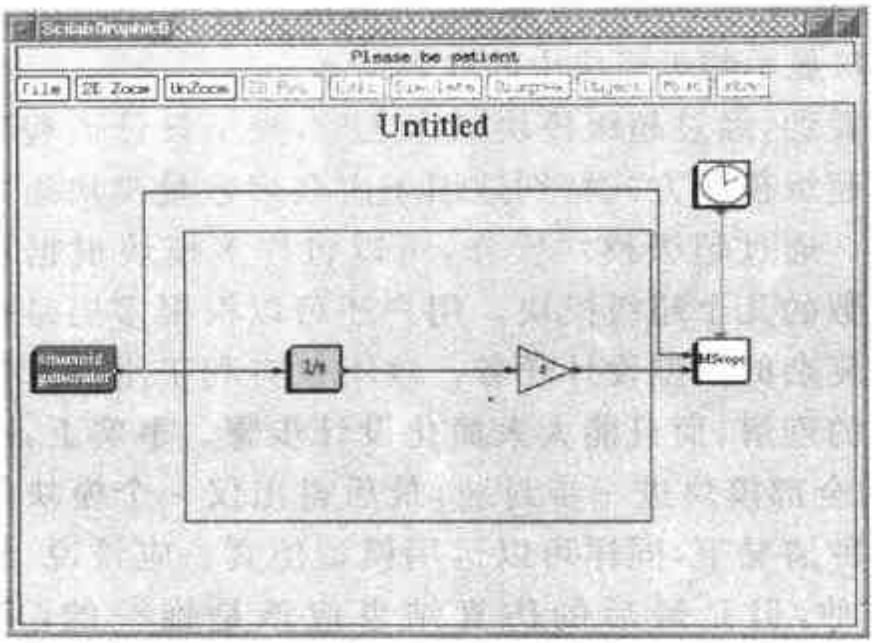


图 8.9 应用"Region to Supper Block"操作

③ Scicos 将区域范围内全部模块自动生成为一个超级模块，如图 8.10 所示。

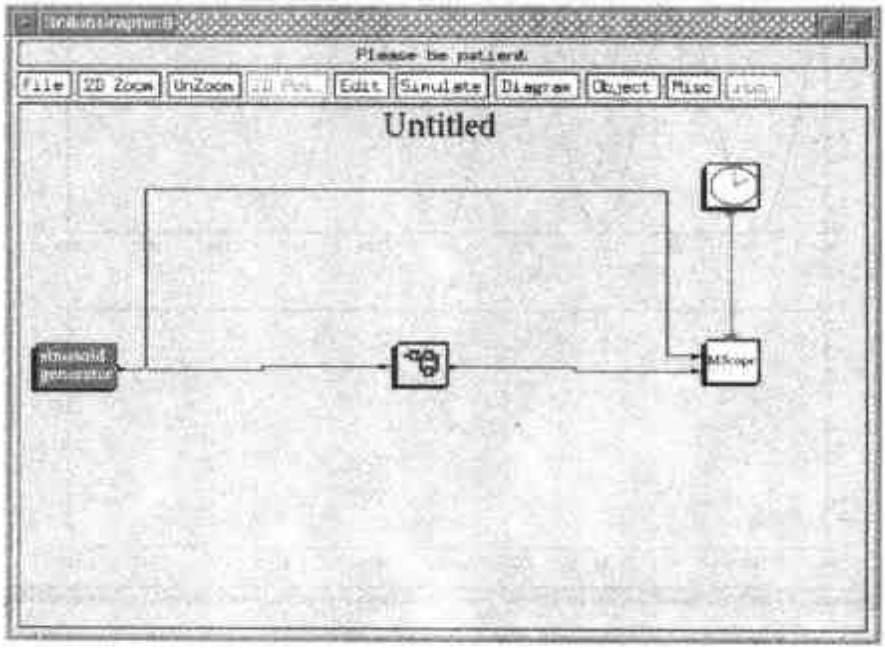


图 8.10 经过超级模块操作后的结果

① 如果希望了解超级模块，可以单击目标，Scicos 将自动弹出一个窗口显示超级模块中的内容。

可以看到，经过超级模块操作之后，模型设计流程图更为简洁。因此超级模块方式特别适用于由众多数量模块组成的动态系统应用。通过超级模块操作，可以将许多模块根据功能分类封装为少数的几个超级模块。用户还可以根据多层结构模块化方式完成复杂的模型设计任务。这不仅有利于用户对物理模型及数据流的理解，而且能大大简化设计步骤。事实上，可以对图 8.10 中的全部模块进一步封装，最后得出仅一个模块的模型设计。在这种情况下，同样可以运用模型仿真。应该说，封装的方式可以多种，但是最后的仿真结果应该是惟一的，应该得到图 8.11 的结果。

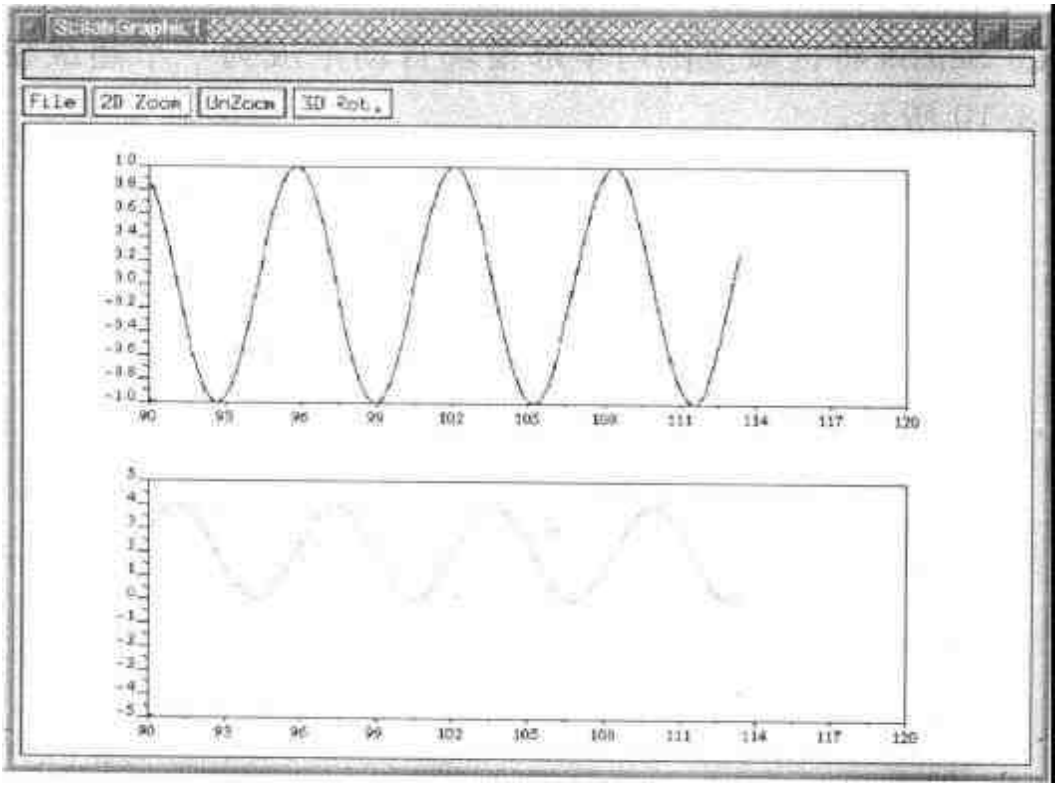


图 8.11 数字信号示波器显示仿真结果

例 8.3 用户自建 SCILAB 函数模块

Scicos 包括一些自建函数模块的应用。函数形式有 C，FORTRAN 以及 SCILAB 语言。在这个例题中讲解自建 SCILAB 函数模块的步骤，其基本步骤与其他语言方式的步骤是相同的。下面是具体步骤：

① 根据图 8.12 的方式选择各个模块，它们分别为正弦信号发生器、增益单元模块、两个 SCILAB 函数模块、多通道数字信号合成器、数字信号示波器和事件信号发生器。

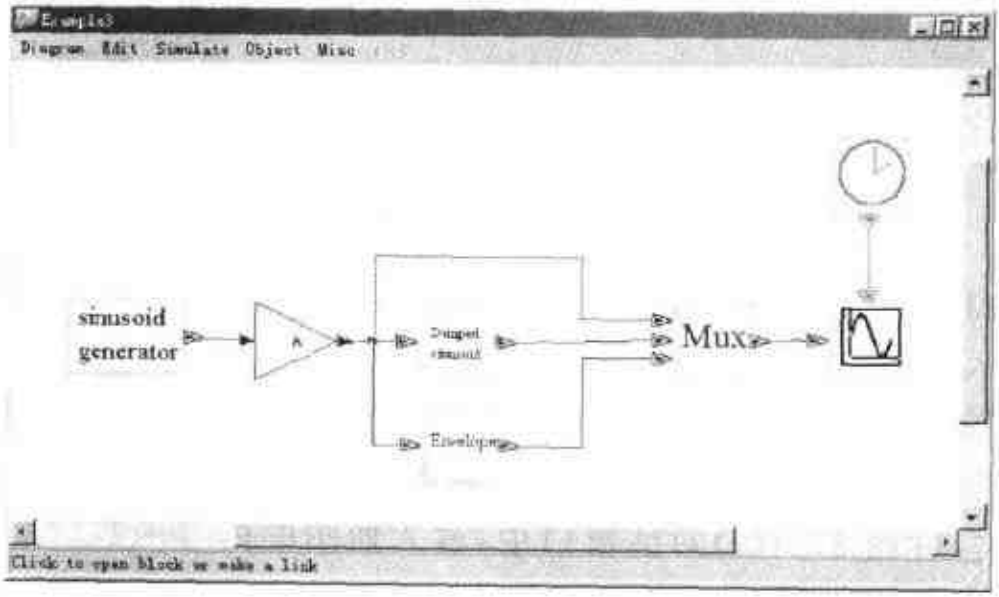


图 8.12 例 8.3 的 Scicos 模型 I

② 在【Edit】菜单中，选择“Context”，Scicos 自动弹出一个对话窗口。按图 8.13 的方式输入模型参数，其中，A 是增益器的增益值；B 是指数函数中的参数值。用该方式设定参数值简单、明了，适合于在多个模块中应用。

③ 正弦信号发生器与事件信号发生器中设定值保持不变。

④ 设增益单元模块中设定值为“A”。

⑤ 单击中间通道的 SCILAB 函数模块，Scicos 自动弹出一个

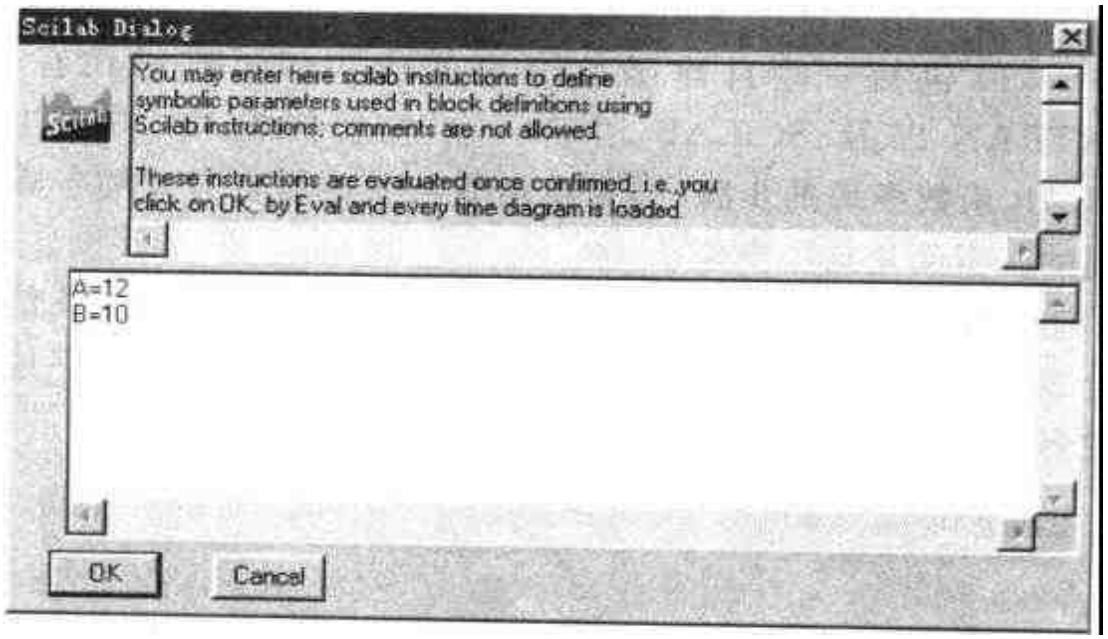


图 8.13 参数设定

对话窗口如图 8.14(a)所示，该设定值不改变，其中“[]”表示“空”值输入。

⑥ 按“下一步”键进入下一个对话窗口如图 8.14(b)所示。不改变设定值，直至进入下一个对话窗口如图 8.14(c)所示。

⑦ 在图 8.14(c)对话窗口中，输入如图中内容函数。其中，t 是事件信号变量；u1 是输入单值变量；y1 是输出单值变量。该函数是指数衰减正弦波形计算。

⑧ 不改变设定值，直至进入下一个对话窗口(见图 8.14(d))。这是输入、输出、状态变量的给定初始值。图 8.14(d)是一个缺省的设置，也可以设为 u1=0; y1=0。

⑨ 下面改变 SCILAB 函数模块名称。选择【Object】中的“Icon”，单击该模块，自动弹出一个对话窗口，在窗口内输入如下语句：

```
xstringb(orig(1),orig(2),['Damped';'sinusoid'],sz(1),
sz(2),'fill');
```

⑩ 用上面的方式，设定另一个通道的 SCILAB 函数模块。这

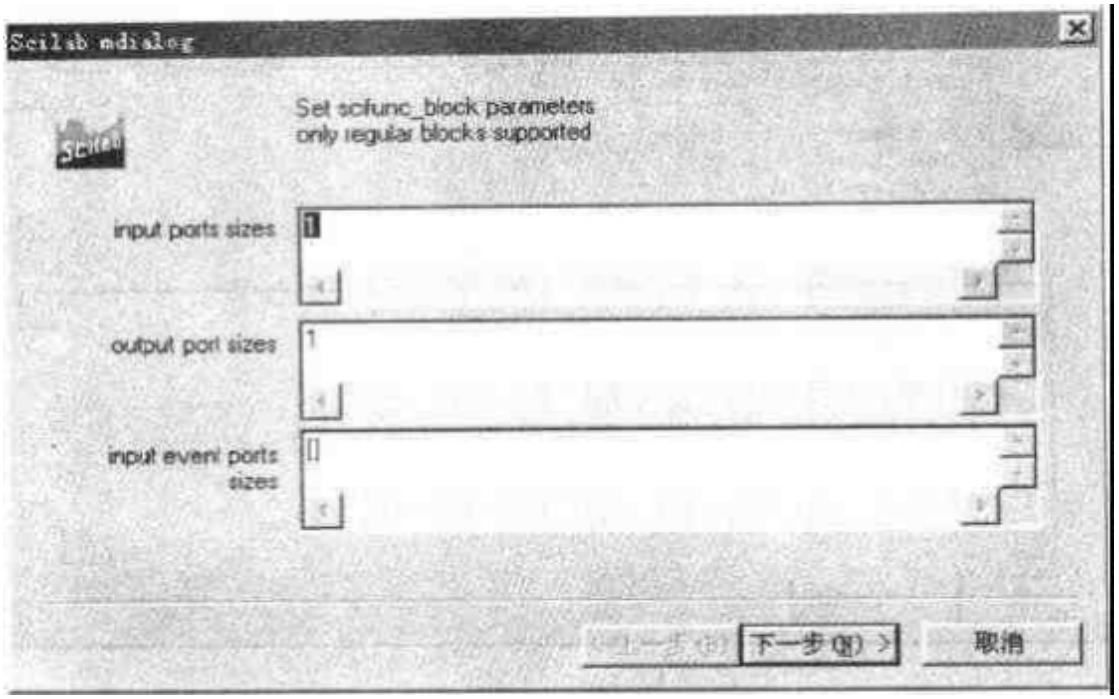


图 8.14(a) SCILAB 函数模块参数设定

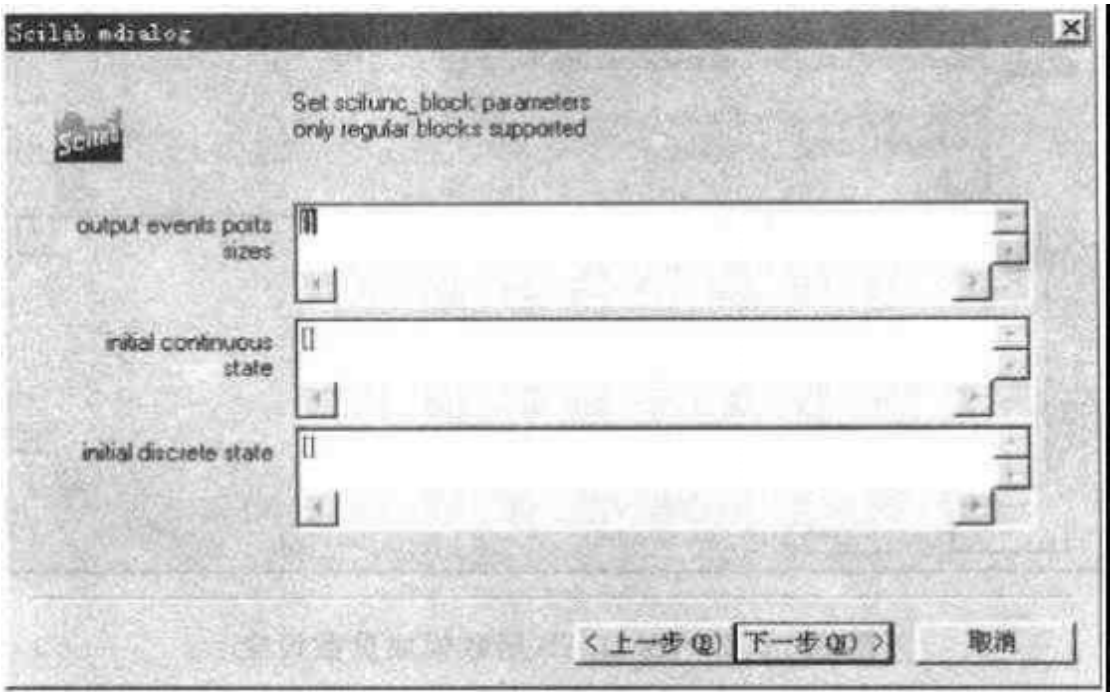


图 8.14(b) SCILAB 函数模块参数设定

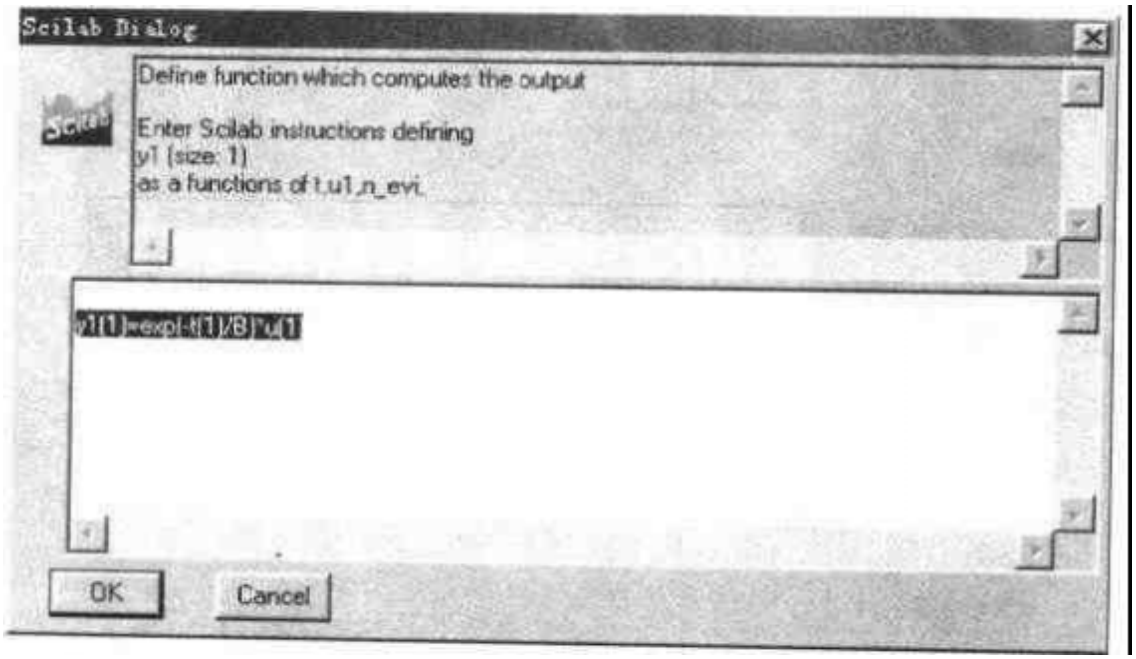


图 8.14(c) SCILAB 函数模块参数设定

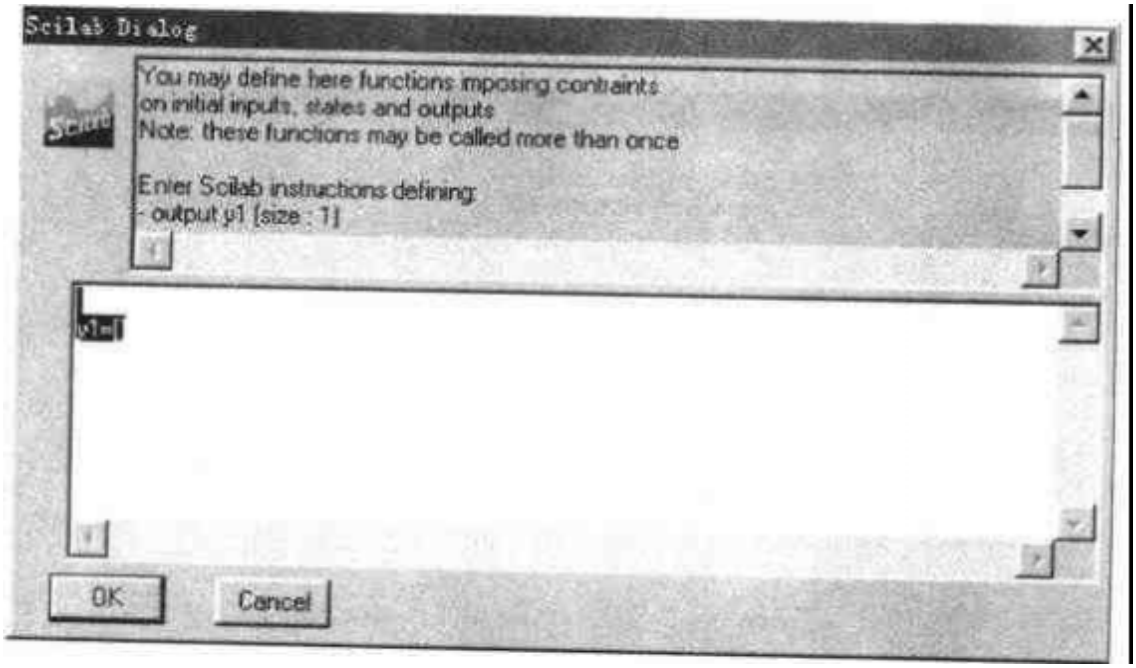


图 8.14(d) SCILAB 函数模块参数设定

个模块是生成指数衰减正弦波形的上包络线。其中应用了如下函数：

```
y1(1)=exp(-t(1)/B) * A
```

注意，该函数中有增益因子 A 是由于事件信号变量 t 是不包含增益因子的。这与图8.14(c)中的 u1 含义不同。下面除了改变该模块名称外，其他设定值与上一个 SCILAB 函数模块的设定值相同。

⑪ 将多通道数字信号合成器的输入个数设定为“3”。

⑫ 根据图 8.12 完成各模块的连线。注意，当对某些模块进行设定或修改操作时，可能要求当前模块没有与外部的连线。

⑬ 改变仿真设置后运行仿真，得出输出结果，如图 8.15 所示。

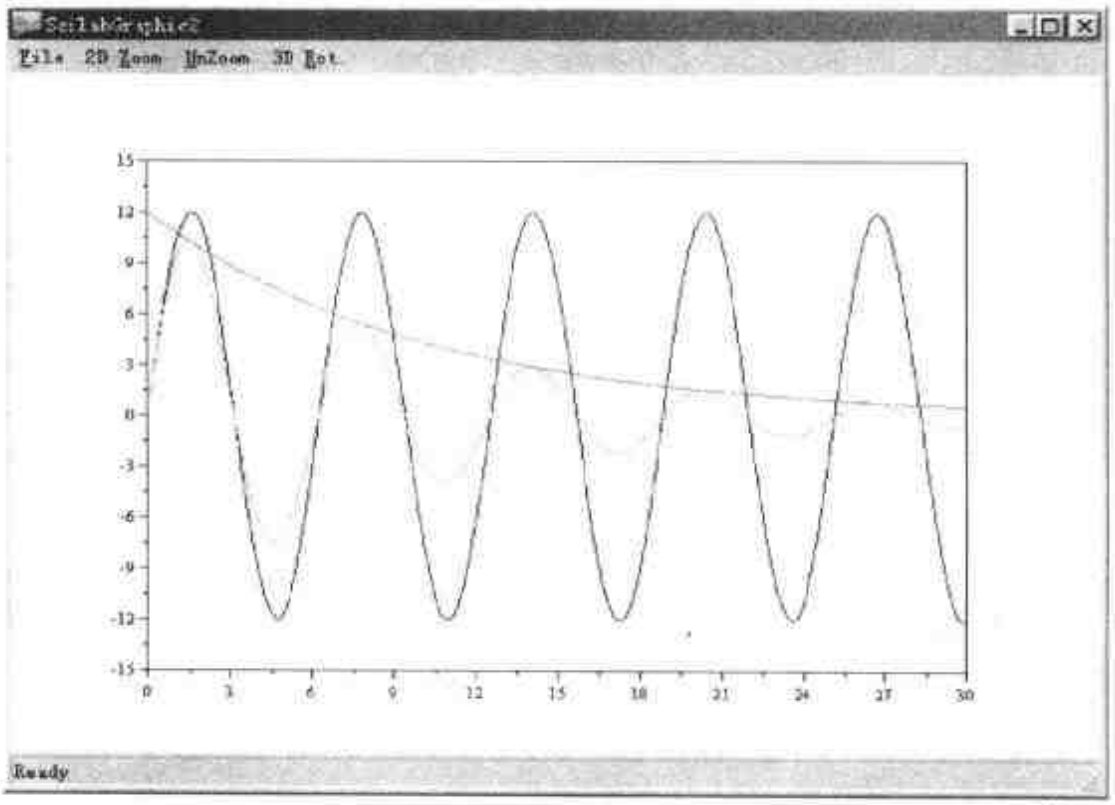


图 8.15 数字信号示波器显示仿真结果

事实上，可以采用另一种简洁的模块方式完成图 8.12 中的功能，如图 8.16 所示，这种方式只采用了一个 SCILAB 函数模块，这

里将该模块改称为“One-three”，其含义是1个单值数字信号输入，3个单值数字信号输出。或者应该理解为是一个向量数字信号输出。下面是一些不同于图8.12的操作步骤：

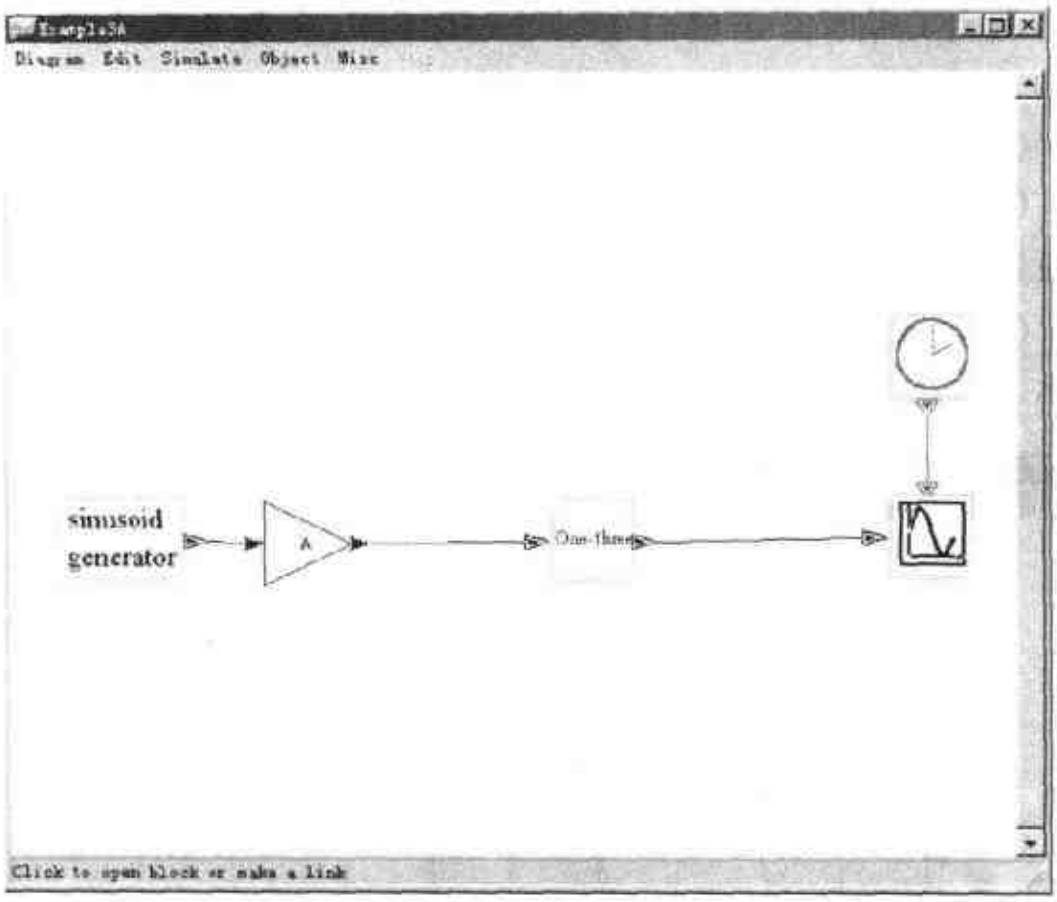


图8.16 例8.3的Scicos模型Ⅱ

① 按图8.17设定各端口个数。此例输入端口个数为1，输出端口为1个3值向量；

② 类似图8.14(c)，按如下语句输入内容：

```
y1(1)=u1
y1(2)=exp(-t(1)/B) * u1
y1(3)=exp(-t(1)/B) * A
```

③ 类似图8.14(d)，按如下语句输入内容：

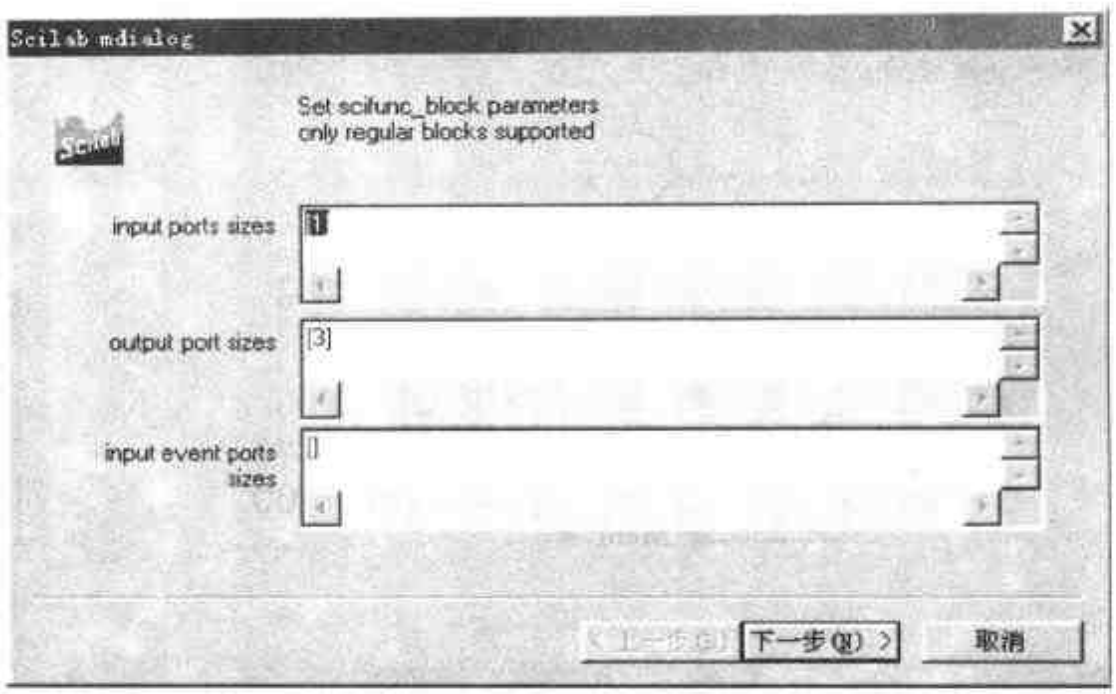


图 8.17 SCILAB 函数模块端口参数设定

y1=[3 : 1]

运行仿真，应该得到与图 8.15 完全一样的结果。

例 8.4 计数器——离散事件模块的应用

此例是最简单的离散事件模块的应用。按照图 8.18 的方式构造计数器。下面是具体步骤：

① 设常数值发生器为“1”。

② 设多通道数字信号加和模块为“[1;1]”，表示是两个通道，为数字信号加和。

③ 设存储器模块中“Initial condition”=0 和“Inherit”= 1 分别表示该模块初始值为零，与该模块外部事件信号进行计算。

④ 名称为“00”模块的是数字显示器。其中改变的设定值有“Font Number” = 1，“Font size” = 11 和“Total number of digits”=4。按照例 8.3 中的步骤改变该模块名称。

⑤ 设事件信号发生器中“Period”= 1。

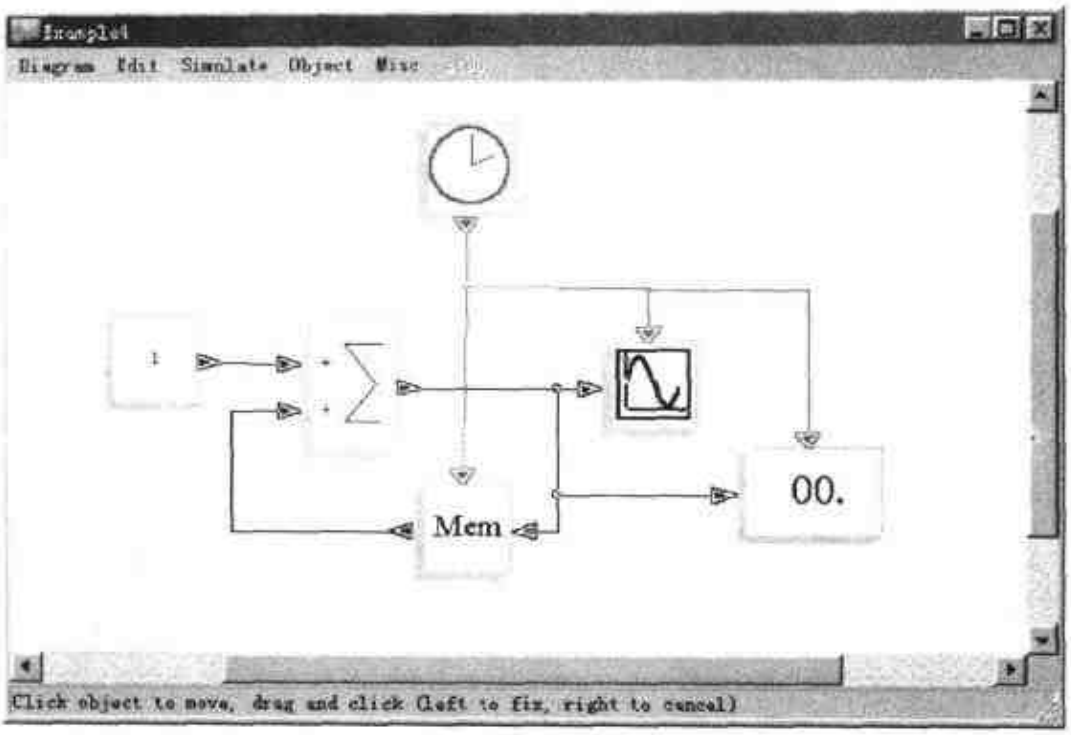


图 8.18 例 8.4 的 Scicos 模型 I

⑥ 单击数字信号示波器，改变“Color and Mark”= −5 3 5 7 9 11 13 15。这是一个有 8 个值的向量。每一个值对应一条示波形式的曲线。当该值大于零时，为颜色输出曲线形式；否则为符号标记的离散点形式。本例中只有一条需要表达的曲线。设定“−5”是输出“0”符号标记的离散点。根据图 8.19 改变数字信号示波器的“x−y”显示范围。

⑦ 进行仿真设置，运行仿真，得到如图 8.19 所示的结果。

可以看到，当有 10 次事件信号触发后，计数器累计结果为“10”。这个结果应该同时可以在数字显示器上看到。

在此例中，要特意介绍一下 Scicos 模块中应用事件内传递(inherit)的功能。在图 8.18 中，将数字显示器与数字信号示波器的两个事件信号连线用【Edit】中的“Delete”方式取消掉，如图 8.20 所示。之后，分别将这两个模块设定值设为“inherit”= 1。

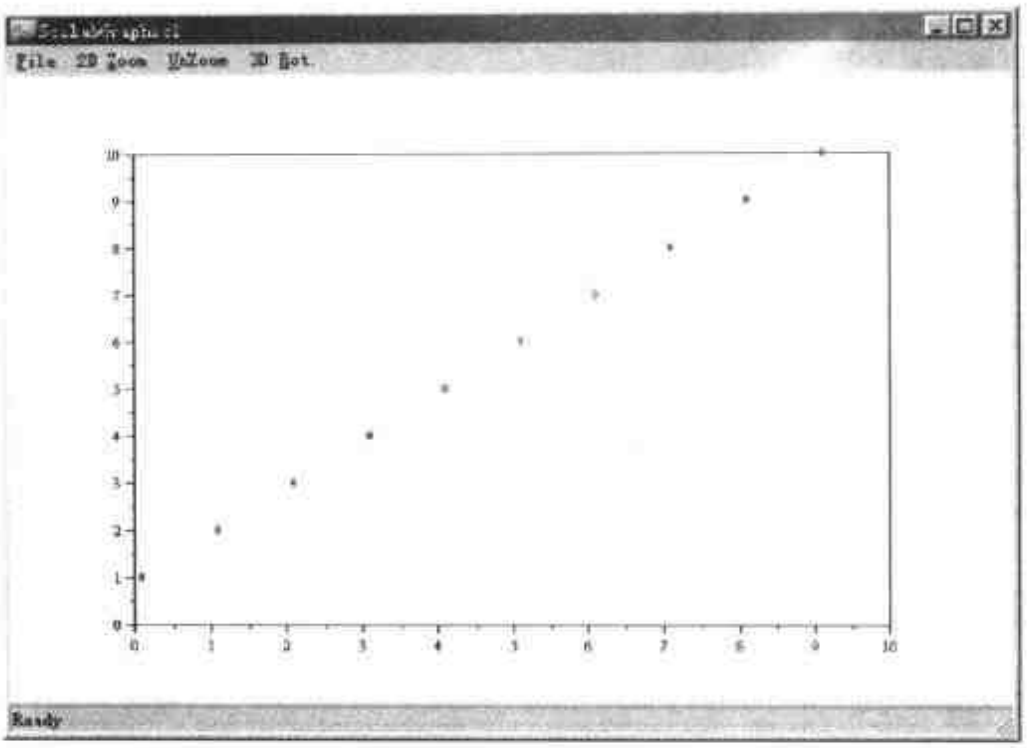


图 8.19 数字信号示波器显示仿真结果

所谓“inherit”意为该模块可以由数字信号通道传递事件信号。在 Scicos 中，事件信号是启动仿真过程的必要信号。根据有否事件信号端口，全部 Scicos 标准模块可以分为两大类：当标准模块上或下方有红色端口时，称它为有事件信号端口模块；否则，称为无事件信号端口模块。对于没有事件信号端口的模块，是根据缺省值“inherit”＝1 的方式运作的，例如该例中的常数值发生器。

对于有事件信号端口的模块，一般可以选择端口个数，这时必须设定“inherit” ＝0。但是，该种模块也可以采用无事件信号端口模式，例如设“inherit”＝1，在此情况下，该模块的前端模块必须有输入的事件信号，如图 8.20 所示的数字显示器与数字信号示波器。而在此例中，存储器不能按事件内传递功能设定，必须要有外部事件发生器的驱动。图 8.20 应该给出与图 8.18 一样的计数结果。

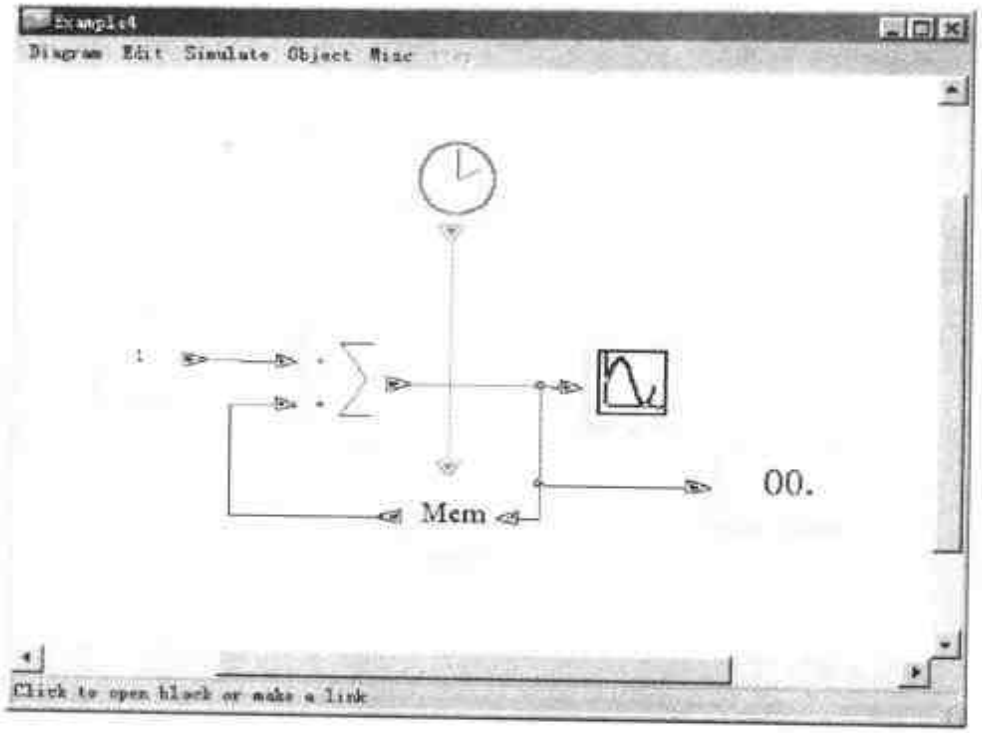


图 8.20 例 8.4 的 Scicos 模型 II

例 8.5 连续系统对象的离散控制

本例题中选用了 SCILAB 演示程序中 Scicos 的第二个例题："连续系统对象的离散控制"。可以从 SCILAB 主窗口中单击下拉菜单的【Demos】项，在弹出的对话选择窗口中选择"Scicos"，然后选择"Continuos plant discrete controller"。SCILAB 将自动弹出 Scicos 主窗口，其中包含 SCILAB 示例设计，如图 8.21 所示。

图 8.21 表示一个一阶连续系统对象应用一个数字离散方式控制器完成正弦信号的跟踪控制。对于这样的一个系统，可以想象被控对象是一个单输入单输出的温度器单元模型，其中，该模型的时间常数是 1s（单击被控对象，由传递函数可以看到该设定值）。此模型忽略了延迟因素，但是在输入端受到了白噪音的干扰。其控制目标是使该温度器单元输出跟踪预定的正弦周期信号。实际应用中可以考虑温度器单元正在实施温度周期疲劳

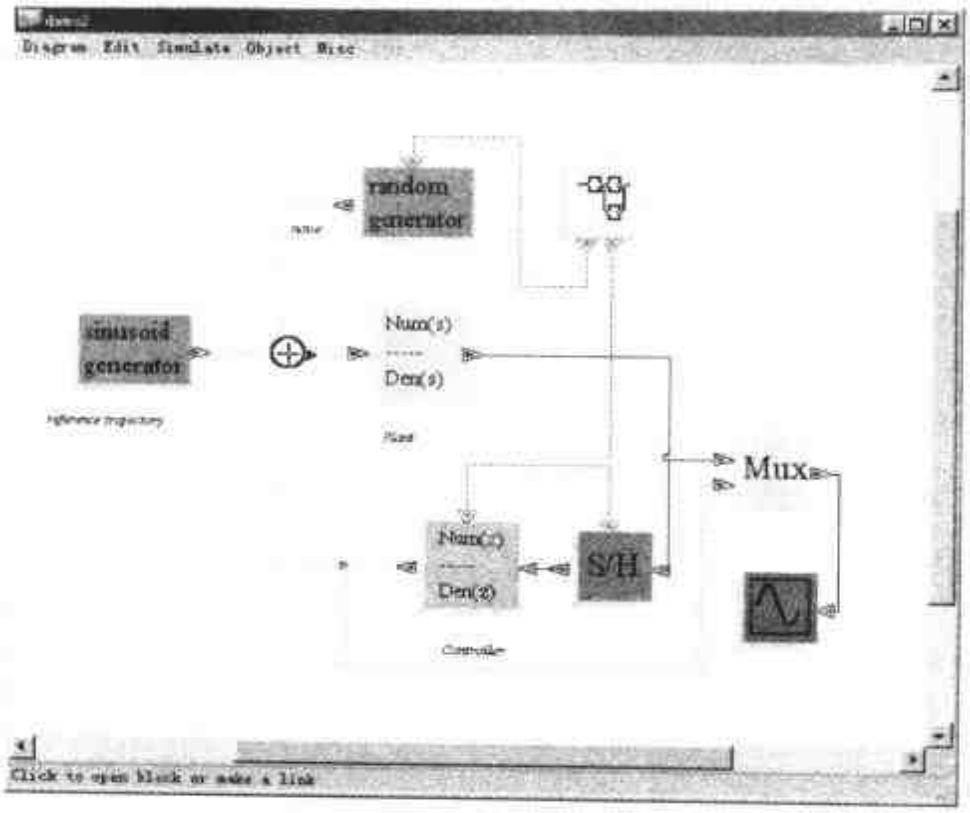


图 8.21 例 8.5 的 Scicos 模型

试验。

从图 8.21 的设计中可以看到，其中的事件信号发生器是由一个超级模块完成的。单击该超级模块，可以获得图 8.22 的窗口显示。这里应用了一个事件时钟分频模块。在超级模块的两个事件输出端口，第二端口是第一端口的 10 分频。这一点可以通过应用上面例题中的计数器来验证。在图 8.21 的超级模块的表示中，左端口对应第一端口；右端口对应第二端口。因此随机信号发生器是以更高的频率刷新随机信号输出，而离散控制器是以低 10 倍的频率计算输出。其中用零阶保持采样单元模块实现连续信号到离散信号的变换。图 8.23 显示模拟控制结果。比较阶梯形状的曲线是控制作用量；而比较连续形状的曲线是控制输出响应。此输出响应对应均值为零，方差为 10 的白噪音输入干扰。如果把方差

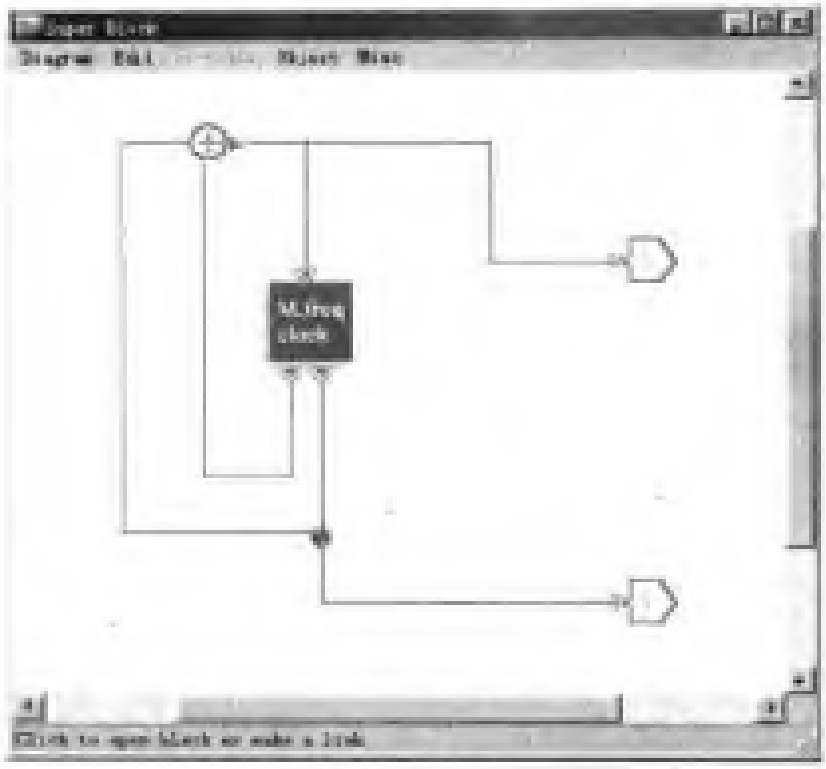


图 8.22 超级模块

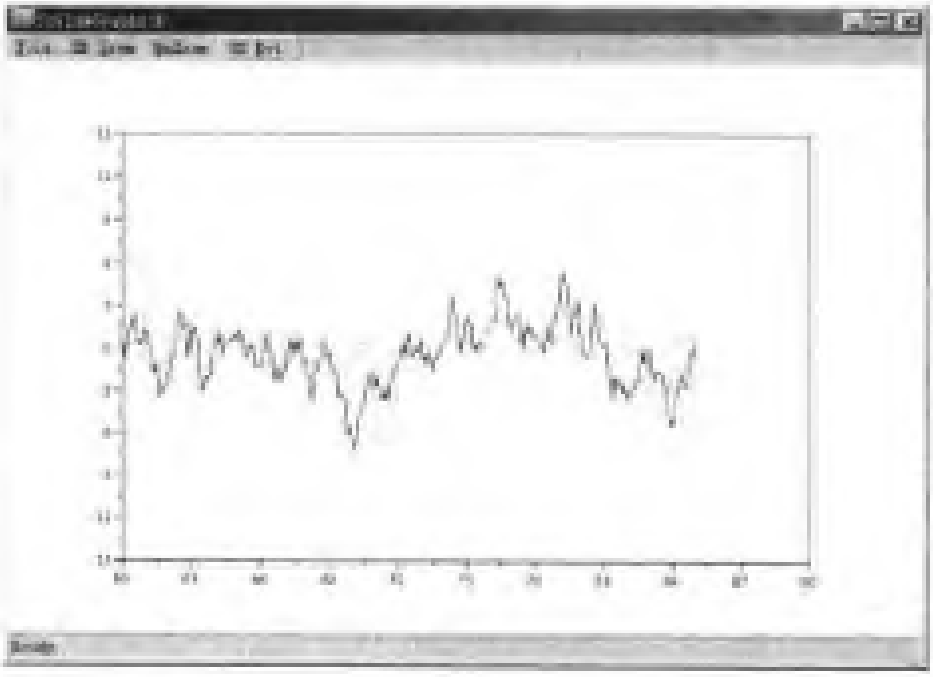


图 8.23 数字信号示波器显示的仿真结果

降低为 2，并改变超级模块中事件时钟分频模块的设定“Basic period”= 0.01，则可以得到较好的正弦周期跟踪曲线。

从此例中可以看到，多通道数字或事件信号加和模块可以完成各通道是不等事件频率的加和功能。在本例中，数字信号为 X1，事件周期为 T1，数字信号为 X2，事件周期为 T2，经加和之后的表达形式为

$$\{X1(t), T1\} + \{X2(t), T2\} = \{X1(t) + X2(t), T1 \cup 2\}$$

其中，加和后的数字信号输出是按照“逻辑或”的方式传递事件信号周期的。

8.6 Scicos 中的模块构造

对任何一个 Scicos 模块的编辑是要经过两方面内容的函数处理：①界面函数用于用户界面处理；②计算函数用于特定功能计算。下面对这两种函数的基本应用做个介绍。

1. 界面函数 (interfacing function)

界面函数一般是用 SCILAB 函数编写完成的。该类函数是存放在 <SCIDIR>/macros/scicos_block 的子目录下。其中，<SCIDIR>代表存放 SCILAB 的根目录，它随用户的存放地址而定。界面函数主要用于确定模块的几何尺寸、颜色、模块端口个数、初始参数、模块标记等方面的信息，它同时还要完成模块中用户对话窗口的信息处理。该类函数的基本语法为

```
[x, y, typ] = block (job, arg1, arg2)
```

其中，输入标记变量“job”决定了该界面函数所要完成的界面处理任务；arg1 与 arg2 分别为模块的数据结构输入；x 与 y 为模块的

数据结构输出；typ 指明端口类型。每一个 SCILAB 模块是用以下形式的数据结构描述的：

list ('Block', graphics, model, unused, GUI_function)

有关更具体的应用界面函数的内容请参见 Nikoukhah 与 Steer 编写的资料，"Scicos—A dynamic system builder and simulator: User Guide"(Scicos——一个动态系统仿真器)，该文是用英文书写的，存放文件名及目录为\C cilab\documentation\pdf\Scicos. pdf，读者可在本书提供的光盘中读取。

2. 计算函数 (computational function)

计算函数既可以用 SCILAB 编写，也可以用 C 或 FORTRAN 语言编写。该类函数是存放在<SCIDIR>/routines/scicos 的子目录下。计算函数将根据模拟器给定的标记完成如表 8.1 所示的工作内容。

表 8.1 计算函数完成的任务与对应的工作标记

标记 (flag)	任务 (task)
0	连续状态导数计算
1	输出值更新
2	状态更新
3	事件输出
4	初始化
5	结束
6	再次初始化

同时，计算函数还将按表 8.2 中的 3 种类型调用函数。其中，变量"eqns"是由界面函数的数据结构中的"model"给定。

表 8.2 Scicos 3 种类型调用函数

函数形式 eqns	SCILAB	FORTRAN	C	说 明
1	否	可	可	调用函数变量可变
2	否	否	可	调用函数变量固定
3	可	否	否	输入输出为 SCILAB 格式

(1) 第一种计算函数类型

当计算函数形式“eqns”为“1”时，计算函数可以使用FORTRAN或C语言。下面给出使用FORTRAN语言的示例。其中模块用了2个输入向量，4个输出向量。

```
subroutine myfun(flag,nevprt,t,xd,x,nx,z,nz,tvec,
&      ntvec,rpar,nrpar,ipar,nipar,u1,nu1,u2,nu2,
&      y1,ny1,y2,ny2,y3,ny3,y4,ny4)
double precision t,xd(*),x(*),z(*),tvec(*),rpar(*)
double precision u1(*),u2(*),y1(*),y2(*),y3(*),y4(*)
integer flag,nevprt,nx,nz,ntvec,nrpar,ipar(*)
integer nipar,nu1,nu2,ny1,ny2,ny3,ny4
```

其中，对各个变量的具体说明由表8.3给出。

表 8.3 第一种类型计算函数的变量说明

输入输出形式	变量名称	说 明
输入	flag	0,1,2, …, 6 (见表 8.1)
输入	nevprt	事件输入码态总数
输入	t	时刻
输出	xd	连续状态导数
输入输出	x	连续状态变量
输入	nx	连续状态变量个数
输入输出	z	离散状态变量
输入	nz	离散状态变量个数
输出	tvec	输出事件时刻(当 flag=3 时)

续表

输入输出形式	变量名称	说　明
输入	ntvec	输出事件端口个数
输入	rpar	实数参数
输入	nrpar	实数参数个数
输入	ipar	整数参数
输入	nipar	整数参数个数
输入	ui	第 i 个输入变量，$i=1,2,\cdots$
输入	nui	输入变量个数
输出	yj	第 j 个输出变量，$j=1,2,\cdots$
输入	nyj	输出变量个数

(2) 第二种计算函数类型

该类型主要使用 C 语言。表 8.4 给出了各个变量的具体说明。下面给出示例：

```
#include "<SCIDIR >/routines/machine.h"
void selector(flag,nevprt,t,xd,x,nx,z,nz,tvec,ntvec,
    rpar,nrpar,ipar,nipar,inptr,insz,nin,outptr,outsz,nout)
integer * flag,* nevprt,* nx,* nz,* ntvec,* nrpar;
integer ipar[],* nipar,insz[],* nin,outsz[],* nout;
double x[],xd[],z[],tvec[],rpar[];
double * inptr[],* outptr[],* t;
```

表 8.4 第二种类型计算函数的变量说明

输入输出形式	变量名称	说　明
输入	* flag	0,1,2,….6（见表 8.1）
输入	* nevprt	事件输入码态总数
输入	* t	时刻
输出	xd	连续状态导数(当 flag=0 时)
输入输出	x	连续状态变量

续表

输入输出形式	变量名称	说　明
输入	* nx	连续状态变量个数
输入输出	z	离散状态变量
输入	* nz	离散状态变量个数
输出	tvec	输出事件时刻(当 flag=3 时)
输出	* ntvec	输出事件端口个数
输入	rpar	实数参数
输入	* nrpar	实数参数个数
输入	ipar	整数参数
输入	* nipar	整数参数个数
输入	inptr	inptr(i)是指向第 i 个输入变量的指针
输入	insz	insz(i)是第 i 个输入变量的个数
输入	* nin	输入端口个数
输出	outptr	outptr(j)是指向第 j 个输出变量的指针
输出	outsz	outsz(j)是第 j 个输出变量的个数
输出	* nout	输出端口个数

(3) 第三种计算函数类型

该类型主要是使用SCILAB语言。表8.5给出了各个变量的具体说明。下面给出编程示例：

```
function [y,x,z,tvec,xd]=test(flag,nevprt,t,x,z,rpar,ipar,u)
y=list();tvec=[];xd=[]
if flag==4 then
    z=0
elseif flag==2 then
    z=z+1
    write(%io(2),'Number of calls:'+string(z))
```

```
    [u1,u2]=u(1:2)
    write(%io(2),'first input');disp(u1)
    write(%io(2),'second input');disp(u2)
end
```

表 8.5 第三种类型计算函数的变量说明

输入输出形式	变量名称	说　明
输入	flag	0,1,2, …, 6 (见表 8.1)
输入	nevprt	事件输入码态总数
输入	t	时刻
输出	x	连续状态向量
输入输出	z	离散状态向量
输入	rpar	实数参数
输入输出	ipar	整数参数
输入	u	u(*i*)是第 *i* 个输入量
输出	y	u(*j*)是第 *j* 个输出量
输出	x	新 x 值,若 flag = 2, 4, 5 或 6
输出	z	新 z 值,若 flag = 2, 4, 5 或 6
输出	xd	x 导数,若 flag = 0
输出	tvec	输出事件时刻,若 flag = 3

第9章

Tcl/Tk 在 SCILAB 中的应用

目前 SCILAB 主要借助 Tcl/Tk 实现用户接口界面功能。Tcl 是 Tool Command Language 的缩写，由美国加州(California)大学伯克利(Berkeley)分校的 John Qusterhout 教授开发。Tcl 主要包含两个部分：脚本语言和用于解释这种语言的解释器。该解释器能够很容易地嵌入应用程序中。Tcl 最大的特点是能够方便地向其他应用程序中添加 Tcl 解释器，扩展其他语言配置，完成定制应用程序的任务。

Tk 是与 Tcl 相关的图形用户界面生成工具包。Tcl 和 Tk 提供了一种可以方便地跨越 UNIX，Windows 和 Macitosh 平台的“虚拟机”。Tcl 虚拟机是可以扩展的，应用程序可以定义新的 Tcl 命令。

Tcl 和 Tk 从 20 世纪 80 年代到目前已经历了多个版本，目前最新的版本是 Tcl/Tk 8.3.4.2。鉴于 Tcl/Tk 解释器的良好移植性和 Tcl 命令的可扩展性，以及都是脚本语言的特点，SCILAB 能很方便地与 Tcl/Tk 相结合，利用 Tcl/Tk 扩展自己的功能。特别是 SCILAB 利用 Tk 能够方便地进行规范的、跨平台的多种用户接口界面的开发。本章首先对 Tcl/Tk 作简要介绍；然后介绍

SCILAB 与 Tcl/Tk 是怎样结合的。

9.1 Tcl 入门

9.1.1 Tcl/Tk 解释器的安装及运行

有许多介绍 Tcl/Tk 的网站，其中 http://www.scriptics.com/software/tcltk/8.4.html 网站内容比较全面，包括 Tcl/Tk 常见问题解答、Tcl/Tk 解释器的免费下载以及一些 Tcl\Tk 源程序代码。从该网站上下载了解释器安装程序，并解压安装后，生成名为 wish.exe 的程序，并且会出现在 Windows 的【开始】菜单中。运行 wish.exe 程序，会出现 Tcl/Tk 解释器的主界面，如图 9.1 所示。

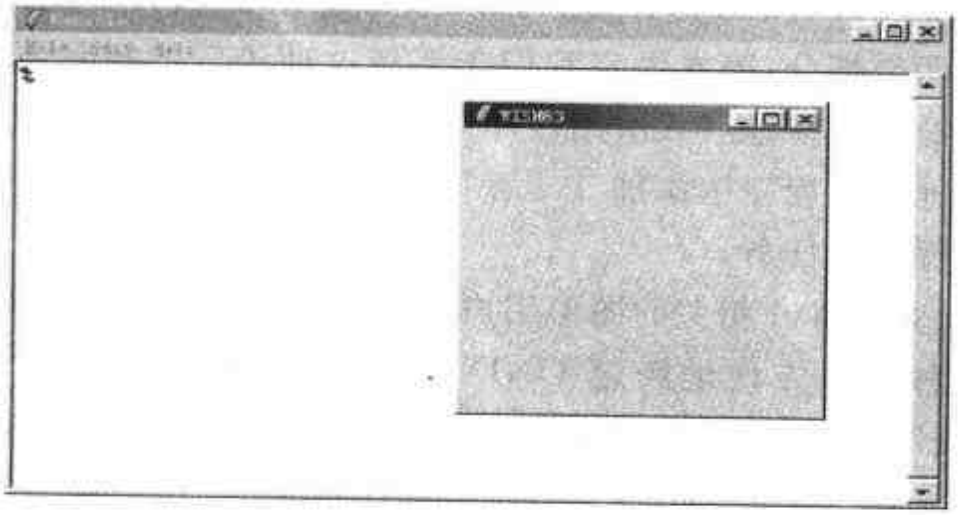


图 9.1 Tcl/Tk 解释器的主界面

主界面包括两个窗口：Console 窗口和 Wish83 窗口。Console 窗口是一个用于交互式输入 Tcl 指令的控制台；Wish83 窗口是 Tcl/Tk 程序的图形窗口。

Tcl 提供帮助文件 ActiveTclHelp.chm，如图 9.2 所示。

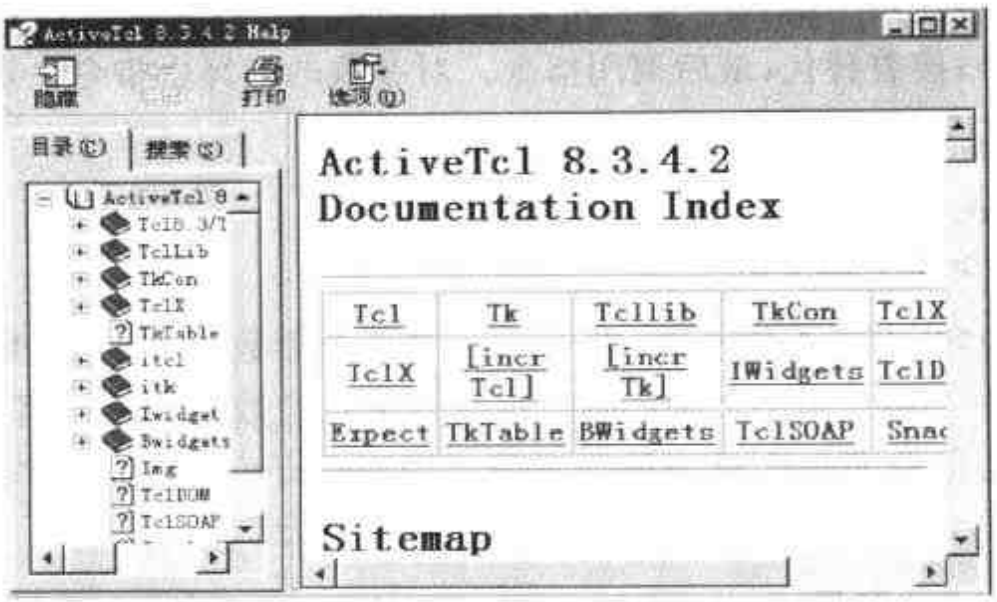


图 9.2 Tcl 帮助文件

9.1.2 Tcl 基本指令

本节简要描述 Tcl 脚本语言的基本语法规则和一些常用的 Tcl 指令，介绍 Tcl 解释器的基本机制，即替代(substitution)和组合(grouping)。

Tcl 是基于字符串的指令语言，只有一些基本的结构和较好的语法，因此简单易学。

1. Tcl 指令格式

Tcl 指令的基本格式如下：

```
Command arg1 arg2 arg3...
```

其中，Command 代表内置指令的名称或者 Tcl 进程；空白区域(即空格和跳格)用来分开指令名称与指令参数；换行和分号用来结束一条指令，指令的参数都是字符串。

Tcl 具有组合和替代两种语法。组合允许一个参数包含多个

单词;替代用于编程变量和潜逃指令的调用。Tcl解释器首先进行组合;接着替代;最后调用指令。对参数的解释由指令决定。

例9.1 输出字符串"How are you?"

```
puts stdout {"How are you?"}
=>How are you?
```

说明:在这个例子中,指令puts有两个参数:一个I/O数据流标识符stdout和一个字符串。花括号不是参数数值的一部分,用来将几个单词组合成为单个参数。"How are you?"是指令puts的第2个参数。Tcl也采用双引号组合字符。

2. Tcl变量

set指令为给变量赋值。该指令有两个参数:第1个是变量名;第2个是数值。变量名可以为任意长度,区分大小写。使用变量前,不需要预先声明变量。解释器会在首次给一个变量赋值时创建它。变量的数值通过在变量名前加符号"$"得到。

例9.2 Tcl变量

```
# How to set varibale
set var 5
=>5
set b $ var
=>5
unset var b; # delete variable
```

说明:Tcl采用字符"#"作为注释标记。Tcl里的"#"必须出现在一条语句的开始,如果将注释附在命令结尾,则必须在#出现之前加分号结束前面的指令。第1个set指令将变量var值设为5,第2个set指令将变量var的数值赋给变量b。利用符号$做替代是Tcl替代的第1种方式。Tcl利用unset指令删除一个变量。允许传递任意数量的变量名给unset指令。

3. 指令替代

Tcl 替代的第 2 种方式是指令替代，利用方括号“[]”将指令分隔成嵌套指令。Tcl 解释器接受括号之间的所有内容，将其当成一个指令。通过替代方括号及其内容来作为外部指令的一个参数，形成嵌套指令。

例 9.3　指令替代

```
set len [string length xing]
=>4
```

说明：在本例中，嵌套指令为 string length xing。该指令返回字符串 xing 的长度。因此本例中的指令相当于 set len 4。

4. 数学表达式

Tcl 解释器本身并不计算数学表达式，而是利用指令 expr 来处理。解释器像对待其他指令一样对待 expr，将解析表达式交给 expr 执行。expr 支持的数学表达式语法与 C 语言相同。指令 expr 可以处理整数、浮点数和布尔值等数据类型。

例 9.4　一个简单的数学计算

```
expr 7.5/5
=>1.5
```

例 9.5　嵌套指令

```
set len [expr [string length xing] + 7]
=>11
```

说明：上述指令 set len 中的第一个变量是以嵌套指令 expr [string length xing]出现的。利用嵌套指令 expr 给字符串 xing 的长度加 7。由于内层的指令替代，由指令 expr 得到 4＋7，因而 len 等于 11。

例 9.6 内置数学函数的应用

```
set pi [expr 2 * acos(0.0)]
=>3.14159265359
```

说明：表达式求值程序支持大量的内置数学函数。

5. 花括号和双引号的组合

双引号和花括号用来将单词组合到一起，形成一个参数。双引号和花括号的差别在于：双引号允许组内替代；而花括号不允许。

例 9.7 双引号和花括号的组合比较

```
set s hello
=>hello
puts stdout "The length of $s is [string length $s]."
=>The length of hello is 5
puts stdout {The length of $s is [string length $s].}
=>The length of $s is [string length $s].
```

说明：本例的第 2 个指令中，Tcl 解释器对指令 puts 的第 2 个参数进行了变量和指令替代。第 3 个命令没有替代，字符串保持原形。

6. 过程

Tcl 利用 proc 定义一个过程。过程被定义后，使用方法就像内置的 Tcl 指令一样。定义过程的基本格式如下：

```
proc name arglist body
```

其中，第 1 个参数 name 是被定义过程的名称；第 2 个参数 arglist 是该过程的参数表；第 3 个参数 body 是指令正文，由一个或多个 Tcl 指令构成。

过程名区分大小写。过程名与变量名不冲突。

例 9.8 过程的定义

```
proc Add{a b} {
    set c [expr $a + $b]
    return $c
}
puts "The sum of 3 and 4 is [Add 3 4]"
=>The sum of 3 and 4 is 7
```

说明:本例中定义的 Add 过程是两个数的简单相加,参数表与指令正文在{}中。变量 c 是过程的局部变量,只有在执行 Add 过程中才有定义;return 指令返回过程的结果,这里 return 指令可以不要,因为 Tcl 解释器缺省返回过程主题的最后一个指令的数值。于是该过程可以精简为

```
proc Add{a b} {
    set c [expr $a + $b]
}
```

注意本例中花括号的书写格式。第 1 行的花括号是指令 proc 的第 3 个参数的开始,即指令正文。在这种情况下,Tcl 解释器一看见打开的左括号,就忽略换行符,扫描文本,直到发现配套的另一个括号为止。

9.1.3 Tcl 基本控制结构指令

同其他编程语言一样,Tcl 也有一些控制结构指令。本节介绍几个常用的 Tcl 基本控制结构指令。

1. If Then Else 指令

if 指令是基本的条件指令。如果表达式为真,则执行一个指令体;否则,执行另一个指令体。第 2 个指令体是可选的。其基本格式如下:

```
if boolean then
  body1
else
  body2
```

其中,关键字 Then 和 Else 是可选的,布尔表达式 boolean 在{}中。实际上,通常省略 Then,而在指令体左右使用花括号。

例 9.9 条件 if then else 指令

```
if{ $ x == 0} {
    puts stdout "hello world!"
} else {
    puts stdout "good news!"
}
```

2. while 指令

while 指令有两个参数:一个布尔表达式和一个指令体,其基本格式如下:

```
while booleanExpr body
```

while 指令反复测试布尔表达式,如果表达式为真(非零),则执行指令体。

例 9.10 while 循环计算阶乘

```
proc Factorial {x} {
    set i 1
    set product 1
    while { $ i <= $ x} {
        set product [expr $ product * $ i]
        incr i;          #i加1
    }
    return $ product
    }
```

调用过程：

```
Fractorial 10
=>3628800
```

3. for 指令

for 指令类似于 C 语言的 for 语句。其基本格式如下：

for initial test final body

其中，initial 是初始化循环的指令；test 是一个决定循环是否终止的布尔表达式；final body 是在循环体中执行的指令。

例 9.11 for 循环

```
set j 0
for {set i 0} { $ i <10} {incr i 3} {
    set j [expr  $ j +  $ i]
}
set j    # 显示变量 j 的值
=>18
```

4. foreach 指令

foreach 指令也是循环指令。其基本格式如下：

foreach loopVar valueList commandBody

其中，loop Var 是变量名；valueList 是第一个参数的取值列表；command Body 是一个指令循环体。该指令用取值列表中的每个变量值循环替代循环体中的变量。

例 9.12 foreach 循环

```
set i 1
foreach result {2 4 6 8 } {
    set i [expr  $ i *  $ result]
}
set i
=>384
```

9.2 Tk 简介

Tk 是一个创建图形用户界面的工具包。Tk 是可移植的，可以在没有改变用户界面代码的情况下，在 UNIX，Windows 和 Macintosh 上运行。Tk 添加了大约 35 个 Tcl 指令，可使用户在图形用户界面里创建和操作部件(widget)。一个部件是图形用户界面的一个具有特定外观和行为的窗体。部件类型包括：按钮、滚动条、菜单、文本窗体以及画布。

Tk 部件组织为一个层次体系。对于一个应用程序，窗体体系一般包括一个主窗体，主窗体内部可能有许多子窗体，子窗体可以有更多的窗体，如此等等。该层次结构会影响 Tk 部件的命名方案。

Tk 利用一些称作“几何管理器(geometry manger)”的指令操作来完成对图形界面中的有关部件在屏幕上的大小和位置的设定。Tk 共有 3 个不同的几何管理器：grid，pack 以及 place。这些几何管理器主要用框架部件作为其他部件的容器，创建漂亮的屏幕布局。

像多数窗体系统工具一样，基于 Tk 的应用程序具备事件驱动控制流。Tk 部件可自动处理多数事件，因此编程很简单。对于特殊行为，可使用 bind 指令绑定事件发生时要运行的 Tcl 指令。

利用 Tk 编程的基本过程是：首先创建部件，使用几何管理器安排这些部件的位置；然后将行为绑定到部件上。解释器处理了初始化界面的指令后，进入事件循环，运行应用程序。

例 9.13 一个简单的 Tk 程序示例(见图 9.3)

```
#! /usr/local/bin/wish
button .hello -text "Hello Everyone!" \
    -command { puts stdout "Hello Everyone!"}
pack .hello -padx 20 -pady 10
```

图 9.3 一个简单的 Tk 程序示例

说明:程序的第 1 行 #! /usr/local/bin/wish 标识脚本的解释器。

在本例中,有两个 Tcl 指令:一个是 button,为创建按钮指令;另一个是 stdout,为使按钮在显示器上可见指令。".hello"为按钮名称;"Hello Everyone"是按钮上的标签;符号"\"是换行符号;pack 指令将按钮投影到屏幕上;-padx 和-pady 参数确定按钮四周空白区域的大小。

9.3 SCILAB 与 Tcl/Tk 的结合

在 SCILAB 中有几个用于与 Tcl/Tk 结合的特定指令。

9.3.1 Tk_EvalFile 指令

该指令用于读取和执行 Tcl/Tk 文件。其基本格式如下:

```
TK_EvalFile(filename)
```

其中,参数 filename 是所要读取和执行的 Tcl/Tk 文件名。

9.3.2 SCILABEval 指令

该指令用 SCILAB 解释器执行 Tcl/Tk 字符串。其基本格式如下:

```
SCILABEval str
```

其中,参数 str 是所要执行的字符串。

例 9.14 SCILAB 与 Tcl/Tk 的结合

首先将下面的代码存入一个名为 test.tcl 的文件中:

```
//test.tcl
toplevel .w1
button .w1.b -text "Click here to see a new SCILAB Graphic Window"\
    -command {SCILABEval "xselect()"}
pack .w1.b
```

然后在 SCILAB 中,用下面的指令执行这个文件:

```
TK_EvalFile(TMPDIR+'/test.tcl')
```

结果如图 9.4 所示。

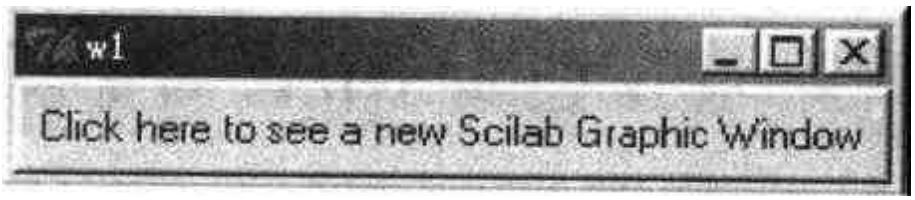


图 9.4 SCILAB 与 Tcl/Tk 的结合示例

说明:该过程在 SCILAB 环境下执行。Tk 指令 toplevel 定义和创建顶层的窗口 w1;button 指令在窗口 w1 中创建按钮部件 w1.b,该按钮的标签是"Click here to see a new SCILAB Graphic

Window";按下该按钮,利用指令 SCILABEval 执行 SCILAB 指令 xselect()。这样,每次按下按钮,就会打开一个 SCILAB 图形窗口。

9.3.3 Tk_GetVar 指令

该指令得到一个 Tcl/Tk 变量值。其基本格式如下:

```
value = Tk_GetVar(varname)
```

其中,参数 varname 是一个字符串,用于保存 Tcl/Tk 变量名;参数 value 也是一个字符串,用于保存 Tcl/Tk 变量 varname 的值。

例 9.15 Tk_GetVar 指令的应用(见图 9.5)

```
Tk_EvalStr('toplevel .tst1');  // 建立一个顶层的 Tk 窗口
Tk_EvalStr('entry .tst1.e-textvariable tvar');
                               // 设定一个编辑框,并确定其对应变量
Tk_EvalStr('set tvar foobar'); // 设置编辑框变量值
Tk_EvalStr('pack .tst1.e');    // 确定编辑框的位置,并显示
text=TK_GetVar('tvar');        // 得到变量 tvar 的值,并返回到 text
```

运行结果如下:text = 'foobar'

图 9.5 Tk_GetVar 指令的应用示例

9.3.4 Tk_SetVar 指令

该指令设置一个 Tcl/Tk 变量值。其基本格式如下:

```
Tk_SetVar(varname, value)
```

其中，参数 varname 是用于设置 Tcl/Tk 值的变量名；参数 value 也是一个字符串，用于保存设置在 Tcl/Tk 变量的值。

例 9.16 Tk_SetVar 命令的应用

```
Tk_EvalStr('toplevel .tst2');                    //建立一个顶层的 Tk 窗口
Tk_EvalStr('label .tst2.l -textvariable tvar');
                                                 // 建立一个静态文本框，
                                                 // 并确定其对应变量
Tk_EvalStr('pack .tst2.l');                      // 确定文本框的位置，
                                                 // 并显示
Tk_SetVar('tvar','This text has been set');      //设置文本框变量值
```

9.4 一个应用实例：greenlab

本节给出一个比较完整的 SCILAB 与 Tcl/Tk 结合应用的例子。该例子应用 Tk 为一个名为“greenlab.sci”的 SCILAB 应用程序建立了一个用户接口窗口界面。在该窗口中，包含一个编辑框，用于输入变量值，并且将该值传递到 SCILAB 应用程序中；还包括两个按钮，一个用于运行 SCILAB 应用程序，另一个用于退出程序。在该例中，还演示怎样建立 Tk 菜单，以及怎样应用 Tk 调用图像。程序的运行结果如图 9.6 所示。

#建立一个主窗口，命名为 greenlab

```
set w .greenlab
catch"{destroy $w}
toplevel $w
wm title $w "GreenLab_LIAMA"          // 主窗口的标题定为
                                      // GreenLab_LIAMA
wm iconname $w "GreenLab"
```

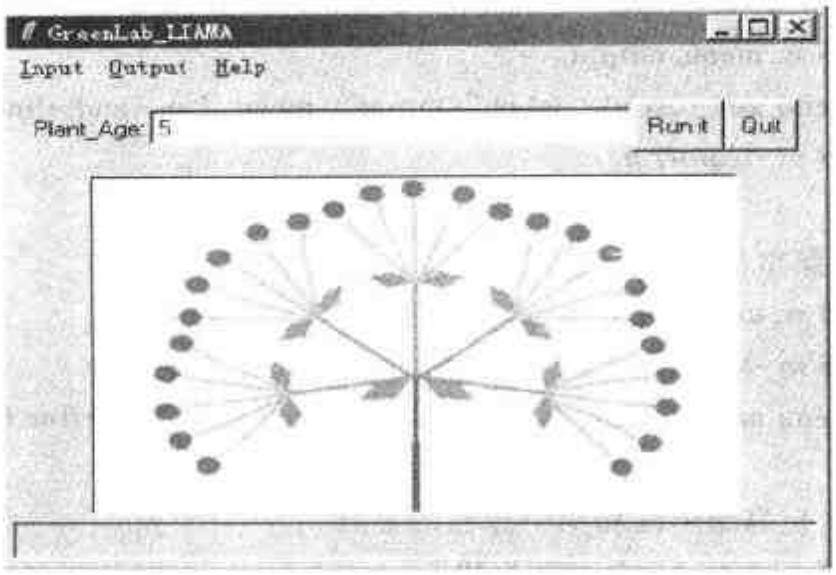


图 9.6 greenlab 实例中的用户窗口界面

```
// 设置一个 statusbar,作为包容菜单的容器
set statusbar "      "
frame  $ w. statusBar
label  $ w. statusBar. label -textvariable statusbar -relief sunken -bd 1 -font
"Helvetica 10" -anchor w
pack  $ w. statusBar. label -side left -padx 2 -expand yes -fill both
pack  $ w. statusBar -side bottom -fill x -pady 2

// 定义一个菜单
menu  $ w. menu -tearoff 0

//定义菜单项 input
set m  $ w. menu. input
menu  $ m -tearoff 0
 $ w. menu add cascade -label "Input" -menu  $ m -underline 0
//定义各菜单项
 $ m add command -label "New" -command {new_game  $ c}
 $ m add command -label "Load para" -command {replay_game  $ c}
```

```
//定义菜单项 output
set m $w.menu.output
$w.menu add cascade -label "Output" -menu $m -underline 0
menu $m -tearoff 0

//定义菜单项 help
set m $w.menu.help
menu $m -tearoff 0
$w.menu add cascade -label "Help" -menu $m -underline 0

// 定义包容编辑框和两个按钮的容器
frame $w.top -borderwidth 10
pack $w.top -side top -fill x

// 定义一个编辑框
label $w.top.l -text Plant_Age: -padx 0
pack $w.top.l -side left
entry $w.top.cmd -width 20 -relief sunken \
    -textvariable command
pack $w.top.cmd -side left -fill x -expand true

// 定义两个按钮
button $w.top.quit -text Quit -command exit
set but [button $w.top.run -text "Run it" -command Run]
pack $w.top.quit $w.top.run -side right

// 将"Run it"绑定于按键 Return 和过程 Run,即按下 Run 按键执行过
// 程 Run
bind $w.top.cmd <Return>Run

// 将焦点集中在编辑框
focus $w.top.cmd
```

```
// 调入一幅图片 greenlogo.gif
set ttt   "C:\\windows\\desktop\\greenlab\\"
catch {image delete image1a}
image create photo image1a -file [file join $ttt greenlogo.gif]
label $w.l1 -image image1a -bd 1 -relief sunken
pack $w.l1 -side top -padx .5m -pady .5m

// 定义 Run 过程
proc Run {} {
    SCILABEval "getf('greenlab.sci');"        // 调用 SCILAB 函数
    SCILABEval "gl_main($command);"
// 将编辑框变量值 command 作为一个参数值传递到 SCILAB 指令，
// gl_main 是 greenlab.sci 中的一个指令
}
```

关于自由软件及 SCILAB

10.1 为什么要讨论自由软件

不同于传统商业计算机软件的应用，在介绍 SCILAB 软件时不应回避关于软件发展领域中的新潮流：自由软件。在最近的几年中，这一主题已经引发了从计算机科学家、软件开发人员、计算机用户到软件商家、投资者和政府部门人员越来越多的讨论或争论。有些媒体将它称之为是一种“发端(initiative)”、“运动(movement)”甚至“革命(revolution)”。作为计算机用户、软件开发者或者软件生产厂家，将面临以下疑惑：

“什么是自由软件，它的真实自由度到底有多大？”

“什么是‘Copyleft’，它完全取消了‘Copyright(版权)’吗？”

“个人或团体发布的版权协议(如 GPL)具有法律效应吗？我编写自由软件的版权被他人侵权后怎么办？”

“自由软件都声明没有产品保证书，这是不是说明自由软件产品质量都很低下？”

“当我使用自由软件中出现问题时，我是否可以得到技术咨询或支持？”

“以 LINUX 为代表的自由软件为用户提供了最便宜的系统平台，但是在其上的应用软件太少，没有发展前景？”

“我是否应该使用自由软件，它是否能给我带来长远的益处以至商业利益？”

“自由软件是否形成了对知识产权保护与商业活动的挑战，它是不是一种‘乌托邦’运动？”

……

事实上，回答以上问题并不是那么简单（如第 1 个问题），类似的讨论仍在持续。本章将介绍有关自由软件方面的内容。但是应该说明的是，其中一些内容仅是我们个人的理解，建议读者阅读相关的网站（见本书给出的相关网站 W1～W10）。本书中的讨论目的是希望与读者共同探索、认识自由软件发展的正确方向，从而能够理性地作出自己的决断和创新。

10.2 自由软件的发展简史

可以认为没有自由软件的发展，也就没有今天这么兴旺发达的软件业。虽然人们可能没有意识到自由软件的存在，但是可能已经无数次地使用或间接地使用了自由软件。许多软件公司，包括微软开发的 Windows 系统软件中都应用了开放源码中的技术。众多优秀软件都是来自于自由软件，例如 Apache 网络服务器，应用该软件的全球用户已经超过 50%以上，远远超过包括微软在内的任何竞争对手。一些网页上已有关于自由软件发展历史的介绍（见相关网站 W3～W5）。可将重要历史事件归纳如下：

- 1968 年，第一个 UNIX 版本在贝尔实验室问世，这是由 Ken Thompson 编写的多用户、多任务，可以免费传播的操作系统。它迅速地被大学、研究机构以至软件企业所采纳和应用。

- 1973 年，Vinton Cerf 与 Bob Kahn 编写出 TCP/IP 协议。该协议后来成为互联网的应用基础。
- 1979 年，AT&T 宣布将 UNIX 商业化计划。为此，加州大学伯克利分校发布 BSD（Berkley software distributions）UNIX 版本。该版本被 DEC 和 Sun 公司接受。不久，Sun，DEC，HP 以及 IBM 等共同建立了“开放软件基金会（open software foundation）”。
- 1984 年，Richard Stallman 建立了 GNU 计划，并发布通用公共许可证协议（general public license，GPL）及 Copyleft 应用规则。自由软件基金会（free software foundation）于同年成立。
- 1989 年，芬兰赫尔辛基大学学生 Linus Torvalds 发布 Linux 操作系统。该系统后来采用了 GPL 协议。目前，应用该系统的全球服务器用户已经超过三分之一以上。
- 1998，1999 年，世界上几个重要软件厂商，IBM，HP，Oracle，CA，Corel 等，宣布支持 Linux 产品计划。
- 2001 年 7 月，在美国召开全球范围内的开放源码大会“O'Reilly Open Source Convention”。微软也派代表参加了会议。微软宣布将有限度地实施开放源码软件发展策略。

了解自由软件与商业软件的发展历史，对于探索正确的软件发展模式是具有启迪意义的。事实上，全面、深刻地认识或阐明自由软件的定义、内涵和意义不是一件容易的事情。因为各种版权协议的自由软件模式不下千种。本文将简单介绍目前最常见的 3 种，而有关 SCILAB 版权协议将在本章最后一节中介绍。

（1）BSD 版权协议

该协议最初是由加州大学伯克利分校发布，其中惟一的要求是用户使用时必须注明软件出处及软件的版权持有人。该协议更

为近似于公共软件范畴。用户在满足惟一条件下可以免费地按照任何方式处置该软件，包括其衍生工作可以是封闭源码。例如微软 Windows NT 使用的 IP Stack 内容便是取自于符合 BSD 版权协议的开放源码程序(该程序功能全面，运行稳定)。BSD 版权协议目前已经派生出几种不同版本(例如 OpenBSD，NetBSD 和 FreeBSD)。有的版本放松到连惟一条件也取消了。

(2) GNU GPL 版权协议

GNU GPL 版权协议是由“自由软件基金会”发布，并简称为“通用公共协议(general public license)”。其中包括了 12 项具体条款。该协议的核心思想是“Copyleft”(有关“Copyleft”更具体的解释请看下一节)，即用户可以自由地使用、修改、分发 GPL 协议软件。但是其衍生作品(指包含原有程序全部或部分内容，或对其原有程序进行修改，或翻译成其他语言的作品)必须也是开放源码，并能够让他人自由地使用、修改与分发。或者说，所有衍生作品必须符合 GPL 协议。应用 GPL 版权协议的自由软件可以采用免费或收费方式进行分发。Linux 正是采用了 GPL 协议的一种操作系统软件。

(3) GNU LGPL 版权协议

这也是由“自由软件基金会”发布的版权协议。其中 GPL 的前缀 L 是随历史而发生的改变。早期为“Library(库函数)”，目前采用“Lesser(弱化)一词”，并称为“弱通用公共协议”。其中包括 16 项具体条款。该协议与 GPL 的惟一不同在于：如果用户用衍生方式应用了 LGPL 协议方式的自由软件(一般为库函数)，则该软件部分可以与用户的专有(proprietary)软件合并使用并销售。但是所衍生的部分必须是开放源码和允许被他人自由使用、修改与分发的。LGPL 没有要求衍生工作必须遵循 LGPL 的版权协议。

从上面可以看到，协议不同，其自由度是不同的。BSD 协议的自由度相对最大。GPL 协议基本上是一个只能包容符合

"Copyleft"规则软件的封闭领域。而LGPL协议部分地放松了GPL的限制，使开放源码与专有软件可以有机地结合起来。但是"自由软件基金会"明确提出不建议使用该协议，而提倡使用GPL协议。除了以上协议外，建议软件开发用户还可关注软件公司发布的诸如"Common Public License"(由IBM编写)以及"Sun Industry Standard Source License"(由Sun公司编写)等版权协议。

10.3 自由软件的分类

本书中用了自由软件这一广义的名词。然而实际上包含更为具体的自由软件定义或非自由软件分类(见相关网站W3～W5)：

(1) 自由软件(free software)

一般指允许任何人使用、复制、修改、分发的软件。英文中的"free"是强调"自由(freedom)"，而不是"免费(free)"。因此分发中可以采用免费或收费方式。

(2) 开放源码软件(open source software)

该类软件的程序源码是公开的，并可以自由地获得。一般将开放源码作为自由软件的另一个主要特征；但是两者还不完全相同，因为有些开放源码不能被自由地修改和分发。

(3) 免费软件(freeware)

它通常指那些可以免费使用、分发，但无法进行修改的软件。因为该软件一般不提供源代码，因此此类软件有时不被列入自由软件的范畴。

(4) 共享软件(shareware)

该软件通常采取"先试后买"的商业运作方式，允许用户自由分发和试用。如果用户决定使用该软件，则必须支付费用。该软件一般不提供源码。

(5) 公共域软件(public domain software)

这是没有注明版权持有人的软件。这类软件提供源代码,并允许用户任意处置软件。

(6) 专有软件(proprietary software)

该类软件主要是商业软件。用户是通过付费方式购买使用的版权。用户对软件的任何复制、分发都必须得到软件版权持有人的许可。此外,该类软件通常不提供源码,用户无法对软件进行修改。

以上只是大体的分类。要想了解某一软件的类别,还需要阅读版权持有人发布的版权协议,并注意它的版权属性,这一点对于所谓"自由软件"的用户特别重要。目前,网站上已经提供了一些有关选择版权协议的内容(见相关网站 W7~W8)。

10.4 关于"Copyleft(开权)"的概念

自由软件的倡导者 Richard Stallman 创立的通用公共许可证协议(GPL)的核心内容之一是所谓的"Copyleft"。这是针对"Copyright"提出的。这个术语的发明很有创意并十分巧妙。从英文角度讲,"right(右)"与"left(左)"可以构成一对反义词。"right"又有"权利"一说。"Copyright"就是常说的"版权"。而"left"按英语的过去分词又有"留下"的含义。因此"Copyleft"从字面上可以定义为留下或空出"版权"的要求。任何人不能在其软件上提出"版权"要求。曾有人将"Copyleft"译为"版权所无"。但是用"开权"一词作为中文术语更为简明、准确。它也可以理解为"开放版权"的简称。以下的解释也说明为什么选择这个术语翻译。

"Copyright"确认并保护作者发明或创造作品的所有权,禁止他人对该知识产权的占有和非许可的复制和使用。而"Copyleft"

并非绝对地取消了"Copyright"。"Copyleft"同样包含"Copyright"的声明,确保作者的所有权,但它放弃了标准"Copyright"中规定的其他权利的要求。或者可以理解为"Copyleft"使"Copyright"变得"名存实亡"。用户除了必须明确表明作品或软件的版权持有人外,还可以自由地分发、使用该知识成果,还可以自由地修改它。但是所有在原有作者基础上衍生出来的工作必须同时满足"Copyleft"的要求。举个例子说,Linux 即是一种基于 GPL 协议上要求"Copyleft"的系统平台软件。当用户使用了 Linux 中的部分程序内容,并对其修改开发出一套嵌入式专用软件后,用户必须声明这是使用了 Linux 的新软件。该新软件作者可以成为新版权持有人,但是新软件必须同样满足"Copyleft"的要求。即他人可以自由分发、使用该新软件,而且可以自由地修改它。

"Copyleft(开权)"的提出是有其原因的。20 世纪 80 年代初在美国麻省理工学院做研究工作的 Richard Stallman 不满于一些软件厂家将公共开发的软件成果改为商业版权软件的方式,为了确保软件产品可以持续地与他人共享,他为此提出了"Copyleft(开权)"的自由软件概念与运作规则,并于 1984 年成立了自由软件基金会(free software foundation)。他所发布的 GNU(GNU is not UNIX)计划与 GPL(通用公共许可证协议)是为了追求软件使用的彻底自由与公平。或者说,"Copyleft(开权)"就是针对某些软件厂家借助自由软件的开放性实现"你的是我的,我的还是我的(What's yours is mine, and what's mine is mine)"的行径。所谓自由软件并不是不受约束,按照 Stallman 的说法,没有人请你非得使用 GPL 方式的自由软件不可。但是,一旦你使用了该自由软件,你就不能以其他方式据为己有,你必须得让他人自由使用你所衍生的成果。否则,就请不要使用该自由软件。

"Copyleft"规则如同一把利剑。当基于该规则的 Linux 计算机系统软件在用户中很快发展起来之后,引起了微软公司的恐慌。

2001 年 6 月，微软首席执行官 Steve Ballmer 先生评论："Linux 犹如一种癌症以智力的方式侵害它所触及到的任何东西(Linux is a cancer that attaches itself in an intellectual sense to everything it touches)"。此言一出，顿时在软件业中掀起不小的波澜。事实上，如果用户只是用 Linux 作为平台而运行自行开发的软件，则不受"Copyleft"或 GPL 的任何限制。例如 Corel 公司开发的基于 Linux 平台上的 Wordperfect 文字处理软件，即属于此情况。Corel 公司既没开放源码，还保留原商业运作模式。Ballmer 先生的这种缺乏事实根据的蛮横攻击，理所当然地受到了计算机行业中人们的回击。双方的争论还未终结。但是"Copyleft"这把双刃剑是否过于锋利？它是否因为强调绝对的自由而使其真正的自由度和灵活度有所下降？如何客观、理性、综合地评价"Copyleft"仍然是未有定论的学术课题。

10.5 关于"开放源码"软件

可以认为没有提供开放源码的软件不能被认为是自由软件。事实上，"开放源码"与"自由软件"已经形成两个相似又不同的阵营(例如 Linux 同属于两个阵营)。当前者(开放源码促进会(OSI))更强调软件的实用、商业属性时，后者(自由软件基金会(FSF))则要坚持保护用户使用软件的绝对自由(见上节)。下面，将进一步介绍开放源码软件以及它在软件发展中的重要意义。

"开放源码"软件是指源程序代码向用户公开的软件。事实上，计算机软件业的起步是得益于开放源码软件的。例如许多商业软件是应用 UNIX，FORTRAN，PASCAL 等自由软件开发的。相当多的商业软件前身是"开放源码"软件，例如 MATLAB。但是随着商业软件公司对软件资源独占方式的扩张，人们开始认识到利用封闭源码方式，收取高额版权费用以追求商业利润的作法，

已经严重地阻碍了计算机软件的良性发展与广泛应用。

目前“开放源码”的概念与实践已经得到越来越多计算机业内人士与商家的认同。那么，开放源码能够给我们带来什么样的益处呢？下面归纳出若干要点：

(1) 对用户而言

- 软件成本降低。
- 软件系统对于用户是透明的。
- 提高软件可靠性和质量。
- 增加软件安全性，降低“后门”或“漏洞”的可能性。
- 用户可以自行裁剪、修改、定制软件，以适应实际问题的需求。
- 减少对某个软件提供商的依赖性，增加用户的选择机会。

(2) 对软件开发人员而言

- 可以实现个性的张扬。
- 可以学习、继承他人的工作成果，避免低水平的重复劳动。
- 更广范围地开展协作开发。
- 有利于更快、更好的创新。

(3) 对软件商家而言

- 可以调动全球优秀软件人员。
- 可以最快的速度消除软件“漏洞”。
- 加快软件产品质量升级。
- 大大降低软件开发成本。
- 为用户提供最大的价格性能比产品，提高产品竞争力与市场份额。
- 避免个别商家的垄断。

在谈到以上益处时，不能回避“开放源码”方式以及目前发展过程中出现的以下问题：

① “小农集市”方式的运作使用户缺乏对软件产品的深入了

解和信任；

② 软件文档等相对薄弱；

③ 缺少规范的、即时的技术支持；

④ 没有质量保证书；

⑤ 完全依赖用户承担软件应用中的风险；

⑥ 过多地要求用户在计算机方面的知识和技能；

⑦ 使“知识产权保护”增加了难度；

⑧ 为竞争对手提供了便利。

如何认识、评价“开放源码”的利弊仍然是一个热门话题。微软曾于 2001 年表示：“开放源代码是促进全球软件和信息技术行业迅速发展的生态系统中不可缺少的组成部分”。一些业内人士认为：“资源共享，服务收费”将是软件工业发展的新模式。“开放源码”正是实现这一模式的关键。由于源码开放，任何商业壁垒和市场垄断将自然被打破。当用户可以自由地交流与使用资源时，软件商家将会更加强调提供服务来赢利，例如特殊模块和技术支持等。

“开放源码”对软件工程与管理、经济社会的发展也提出了新课题。对于像操作系统软件这样的复杂系统，如何设计并完善该系统是有很大难度的。“开放源码”基本上是采用“自下而上”的脑力劳动者集市运作方式，它为什么会比“自上而下”的设计方式更为有效？“开放源码”的适用范围是什么？如何处理“知识产权保护”与“开放源码”的关系？可以相信，多元化的版权方式是必要的。在排除软件模式发展讨论中非理性化的争执时，应该将重点放在版权协议技术层面上建设性的研究上。

10.6 关于 SCILAB 版权协议的说明

本节的内容主要是基于作者与法方 SCILAB 研究组部分成员的讨论。目的是为了更明确地了解 SCILAB 版权协议的具体

设计思想，并希望读者较好地掌握并应用该协议的注意事项。但是在此需要特别说明：**当本节内容出现与法文版SCILAB协议有不同之处时，应该以原协议内容为准。**下面是所总结的若干要点，并尽可能给出实例以便理解。

(1) SCILAB版权协议规定版权持有人的目的不是简单地明确版权持有人对软件的发明权，而是要明确该软件的管理权，以保证SCILAB软件的修改、管理、发布等过程中的惟一性。可以设想，没有这种惟一性的约束，将无法控制不同类型SCILAB版本的出现。这对一个开放、自由软件的长期发展是极为不利的，或者是灾难性的。因为分散的、多类型的SCILAB软件版本将使用户面临选择方面的困惑。分散、无序的发展最后可能导致软件的终结。规定版权持有人的根本目的是为了保证SCILAB软件能够健康、有序地向上持续发展。

(2) SCILAB版权协议中规定了3种软件定义，分别为：本软件、SCILAB衍生(derived)软件和SCILAB合成(composite)软件。根据不同的软件定义，协议要求有不同的使用规定(可参见本书附录B的内容)。需要特别注意和理解的是，关于SCILAB合成软件的定义和规定。

(3) SCILAB是目前使用自行发布的专用协议。它既不是GPL，也不是LGPL或BSD等其他形式的协议。在实施商业应用方面上，它比GPL或LGPL更为宽松。但是由于SCILAB合成软件的商业应用需要征得SCILAB版权持有人的同意，它又比BSD协议更为严格。

(4) 任何用户可以采用类似于Linux软件方式的那样免费地复制，并有价地销售SCILAB软件。但是由于互联网上可以随意、免费地下载该软件，因此用户一般需要提供附加值内容，例如通过提供用户手册、学习教程、培训等方法来获得更大的商业利益。

(5) 解释 SCILAB 合成软件定义与规定的最好方式是应用 MuPad 软件实例。MuPad 软件是由德国 MuFace 公司发展的一个用于符号计算的商业软件(但是 MuPad 也有免费使用的教育版本,可以从网上下载)。为了扩充数值计算部分,MuFace 公司计划将 SCILAB 软件的计算功能纳入其 MuPad 新版本之中。为此,SCILAB 相当于一个子模块方式被合成到新版本的 MuPad 软件中,而不再是一个独立的软件系统。用这种方式使用的 SCILAB 整体或部分,被定义为 SCILAB 合成软件。根据 SCILAB 版权协议,新版本的 MuPad 软件中必须有明确的标注:"composite software using SCILAB (c) INRIA-ENPC functionality"。在新版本 MuPad 软件按照商业方式销售前,MuFace 公司必须征得 SCILAB 版权持有人的同意。该 SCILAB 合成软件部分必须保证是开放源码。而 MuPad 软件中的其他部分是由 MuFace 公司自行决定的。

(6) 对于用户自行开发的工具箱、应用软件、用户界面等作为独立的软件在 SCILAB 平台上运行,不属于 SCILAB 版权协议所定义 3 种软件中的任何一种。或者说,这种非合成方式调用 SCILAB 的软件是完全属于用户自主产权的软件,用户可以按照任何方式处置这些软件。例如用户可以采用封闭源码编写,并可规定独有的版权协议。当用户将这些自主产权软件商品化时,不存在需要征求 SCILAB 版权持有人同意的问题。在中国举办的"2002 年 SCILAB 竞赛"中,所有参赛作品都属于这种完全用户自主产权的软件。如果参赛作者愿意商品化自己的软件作品并进行销售,则可以附加上 SCILAB 的全部程序和文档文件,以便用户使用。

(7) 如果将 SCILAB 改为纯中文文档的 SCILAB 软件方式,则新软件将按照 SCILAB 合成软件定义执行。

由以上要点可以看到，SCILAB 版权协议是比较符合“公平、合理、开放”的原则的。虽然在 SCILAB 版权协议中，没有关于 SCILAB 合成软件在商业应用中如何获得 SCILAB 版权持有人同意的基本原则阐述，但是我们被告知，用户是被鼓励按照“公平、合理、开放”的原则对 SCILAB 软件进行商业化使用和二次开发。

第11章

未完成的结束语

科学技术犹如给人类的发展插上了翅膀。在这信息、知识的时代没有任何力量可以阻挡人们自由驰骋的思想。即使你是一个学生，绝不要小看自己的能力。由于年轻人有更少的束缚力，因此软件事业更偏爱于年轻人。创新让你获得快乐，成果分享会给你带来更大的快乐。你可以选择各种各样的创新道路。例如走Linus Torvalds在大学时期开创Linux共享软件发展之路。或者，你可以像中国科学院自动化研究所刘迎健研究员那样，在基于研究生学习基础之上开创中文手写输入的“汉王”软件商业发展之路。

SCILAB为你的创新提供了极好的平台。高中教育中的数学、物理学习即可以借此平台。例如应用图形以至动画方式模拟不同条件下抛射物体的运动轨迹。当物体初始速度给定时，如何选择合适的抛物线与地面夹角以达到最远发射距离；或者已知距离，如何确定夹角；另外，如何将数学实验引入化学与生物课程中去等。这些都会构造出有意思的挑战。应用SCILAB，同学和老师的创新成果可以名正言顺地标注上自主版权。同时，还可以设计出你所希望的版权协议与他人共享或者要求商业利益。

计算机的发展以及自由软件的应用带来的难题不仅是技术方面的，而且更大的挑战在于法律、社会、思想以至哲学等方面。

在法律方面，当你发展的自由软件存在漏洞，他人为你提供了补丁文件。这时，如何完整地体现你和他人的版权；另一方面，个人协议是否具有法律效应？如果是，那么不同协议发生冲突时如何处理？如果不是，它在社会中对他人的约束力靠什么？是简单的“诚信力”么？

在社会方面，Richard Stallman 引用了美国林肯总统关于人权与产权关系的论述：“当人权与产权发生冲突时，人权应该是优先的（Whenever there is a conflict between human rights and property rights, human rights must prevail）。”产权的目的是为了促进人类的发展，而不应该成为社会进步的障碍。在计算机网络时代面临的所谓“数字隔离”问题中，人们应该如何应对人权与产权的矛盾？最为基本的、通用的软件部分是否应该走向自由与共享？如果是，又应该如何作好与商业软件的协调发展？

在思想方面，中国人面临的最大挑战是“尊重知识产权，合法使用他人成果”观念的普及。同时，如何将这一观念变为我们的行为习惯更需要时间。计算机的发展又将深刻地改变人们的生活方式。新的观念，例如“开放源码”、“Copyleft（开权）”、“地球村”、“全球化”等概念已经引人。我们对此是否已经有所准备？例如，“全球化”将怎样改变我们的思维模式？我们将怎样保持个性和多元化？

谈到哲学方面，可能使人感到空洞和遥远。但是试问“计算机硬件按照摩尔定律（Moore's law）的高速发展是否也会给人们的健康发展带来危害？”“计算机的广泛应用是否使人们更加远离大自然？”“人们是否会成为科学技术的奴隶？”“现代社会中，人们对生活幸福的追求目标是什么，又将如何实现？”

我们希望看到读者能够提出更为深刻的问题，虽然面对问题

可能让我们感到沉重。但是能够提出一个好的问题，或者给出有见解的答案，难道不是一件令人愉快的事情吗？以上的话题可能超出了本书的重点。但是，我们强调的是，“开放源码”可以使你在更为广阔的空间中自由飞翔。

最后，让我们回到本书题目“自由软件”用词的选择上来。该名称是由 Richard Stallman 等“计算机侠客”人士建议使用的。然而，该阵营中的 Eric Raymond 后来提出了“开放源码软件”概念，或简称“开源软件”。由此产生了“自由”与“开源”两个阵营。我们理解，两者有相似的内涵，但不等同。从技术观点的角度上看，Raymond 所提的“开源软件”更为合理、实际，而 Stallman 的提法过于理想和偏激。不过，本书中仍然采用“自由软件”一词，这是由于“自由”两字更能表现人们对“自由”渴望的本性追求。我们很高兴地看到 SCILAB 软件建立在理性的基点上，以“开放源码”为方式的“自由、共享、同创”模式将会在科学、教育领域里蓬勃地发展起来，下一个问题将是 SCILAB 未来的用户是否能在这样的一个平台上做出有声有色的工作呢？

我们期待看到您的回答。

附录 A　SCILAB 光盘使用说明

1. 光盘文件的基本构成

本光盘文件的基本结构如下。本附录中将目录名称用黑体字，文件名称用常规字体。

Readme：纯文本文件，中文书写。简介本光盘。

Book：　关于本书《科学计算自由软件——SCILAB 教程》相关资料的主目录（全部内容有<1M）。

Scilab-2.6：包括由 INRIA 与 ENPC 提供的全部 SCILAB 软件及其相应的文档文件（全部内容有 197M）。

2. 使用本光盘内容的要求

我们欢迎读者和用户免费、自由地使用、复制或者传播本光盘中全部内容的软件和资料。但是提醒读者和用户务请注意以下使用要求：

（1）当使用 **Book** 目录下的文档资料时，务请明确、完整地注明文档版权人例如以下格式："（c）LIAMA. All rights reserved."。

（2）当使用 **Scilab-2.6** 目录下的软件与文档资料时，务请阅读 SCILAB 版权协议，并按照协议规定使用。

3. 本书相关资料的文档文件(Book 目录下)

除了"Index"文件外,其他文件全部为 PDF 格式。文件名称及内容如下:

Index: 通过 HTML 格式文件阅读本目录中的全部文档文件

Foreword_A: 序一

Foreword_B: 序二

Preface: 前言

Abstract: 内容简介

Contents: 目录

Appendix_Ⅰ: 附录 A SCILAB 光盘使用说明(即本文)

Appendix_Ⅱ: 附录 B SCILAB 版权协议(中文译本)

Appendix_Ⅲ: 附录 C SCILAB 部分函数指令表

4. SCILAB 软件及文档文件(Scilab-2.6 目录下)

本目录下包括 SCILAB 2.6 版本的软件及其相应的文档文件。下面是有关该目录下的一些子目录及相关文件。当采用自动安装 SCILAB 方式时,其目录内容会有所不同。

Autorun:用于 Windows 平台下 SCILAB 程序安装的自启动。

Contrib:共包含用户提供的 33 个工具箱程序(按子目录存放):

Readme:该子目录简介;

Index:33 个工具箱名单。

demos:SCILAB 应用演示文档文件。

demos:通过该 HTML 文件阅读本目录中全部演示文

档文件。

distribution：包含 SCILAB 2.6 版本在各种平台上的 Winzip 压缩文件。

documentation：SCILAB 软件介绍文档文件（以下 3 个子目录中的内容是等同的）：

html：该子目录包含全部 HTML 格式软件介绍文档文件，其中，doc 为通过该 HTML 文件阅读本目录中全部文档文件；

pdf：该子目录包含全部 PDF 格式软件介绍文档文件，其中，pdf 为通过该 HTML 文件阅读本目录中全部文档文件；

postscript：该子目录包含全部 POSTSCRIPT 格式软件介绍文档文件，将各文件用 WINZIP 打开后由 PS 文档阅读软件阅读。

html：包含 HTML 文档中应用的各种标识(Icons)。

Windows：包含应用在 Windows 平台上的 SCILAB 2.6 版本软件

该目录下的具体内容请见下一章节。

linux：包含应用在 Linux 或 UNIX 平台上的 SCILAB 2.6 版本软件。该目录下的具体内容与 Windows 目录中基本相同，可以参考下一章节。

autorun.inf：用于 Windows 平台下 SCILAB 程序安装自启动。

index.html：通过该 HTML 文件阅读本目录中重要文档文件(法文)。

licence.txt：SCILAB 版权协议(英文文本)。

Readme-UNIX：英文的 readme 文件，UNIX 或 Linux 平台。

Readme-Windows：英文的 readme 文件，Windows 平台

如果用户希望阅读文档资料学习 SCILAB，我们建议用户在 SCILAB 2.6 目录下，单击“index”文件。通过该文件的文本超链接方式，可以阅读“Documentat”与“Demo”子目录下的全部文档文件。这些文件是用英文书写的。

5. Windows 目录下的文件

考虑到中国计算机用户多数用 Windows(NT/2000/95/98)平台。因此主要介绍 Windows 目录下的文件。Linux 目录下的文件与此基本相同。该目录大约有 29.4M 内容，下面是主要的子目录及相关文件。

Bin：可执行文件子目录。SCILAB 的可执行代码文件是 scilex.exe。这个目录还包括用于打印 SCILAB 生成的 Postscript 或 Latex 文件的外壳程序。

conf：配置程序子目录。

contrib：用于存放用户提供的程序。

demos：共包含 30 个子目录的演示程序。这些演示程序代码是由 SCILAB 脚本文件格式写成(但是扩展名是 *.dem)，对初学者很有用。例如，alldems.dem 文件是通过 SCILAB 的主窗口【Demos】下拉菜单调用。

examples：共包含 14 个子目录。包括一些怎样链接外部程序到 SCILAB，或应用动态链接以及 intersci 的演示示例。

imp：用于 SCILAB 打印方面的程序。

macros：共包含 21 个子目录。

man：帮助文档子目录。

maple：与 Maple 软件连接 Maple 程序文件。

routines：C 或 FORTRAN 程序库。

tcl：应用 Tcl/Tk 编程语言的子目录。

tests:测试 SCILAB 用的执行代码文件。

util:用于管理 SCILAB 的 ASCII 文档文件。

ACKNOWLEDGEMENTS:对参与 SCILAB 程序的主要编写人员的致谢。

changes:关于 SCILAB 2.6 版本与 2.5 版本不同的说明。

configure:提供通用的建库服务程序。

Libool:提供通用库支持服务程序。

licence:SCILAB 版权协议(法文版本)。

license:SCILAB 版权协议(英文版本)。

Makefile. incl. mak:包含所有 Makefile 中的文件。

Makefile. mak:生成 Makefile 文件的主要文件。

Readme:关于安装与卸载 SCILAB 软件的文件。

Readme_Windows:关于安装与卸载 SCILAB 软件的文本文件(在 Windows NT/2000/95/98 平台下。

scilab. star:SCILAB 启动后初始化程序,在 SCILAB 主窗口下拉菜单【Control】中选择"Restart"将启动该文件

Version. incl:包含 SCILAB 版本信息,被部分 Makefile 文件调用

6. 阅读本光盘文档文件需要的软件环境

本光盘基本包括 3 种类型文档文件格式。它们分别如下:

纯文档文件格式:*. txt, *. doc, *. 或 *. *

HTML 文件格式: *. html, *. htm

PDF 文件格式:*. pdf

不论何种计算机平台,用户一般都可以阅读前两种格式文件。若用户计算机没有可以阅读 pdf 格式文件的 Acrobat 软件,建议用户在互联网上自行下载并安装免费版的Acrobat软件。

7. 阅读、编辑 SCILAB 脚本文件的 PFE 软件

由于 SCILAB 没有提供专用的脚本文件编辑软件，建议 Windows 平台上的用户从网站上下载相关的自由软件。其中，PFE(programmer's file editor)是一个好的选择。应用该软件可以大大加快文件编辑、阅读的过程。该软件目前的网址如下：

http://www.lancs.ac.uk/people/cpaap/pfe/

附录 B　SCILAB 版权协议(中文译本)

(译者注:本译文的全部条文将以“SCILAB 版权协议”的法文原文为准。该法文原文请见光盘内文件)

1. 前言

制定本版权协议的目的是向 SCILAB 用户明确指出使用、修改、传播本软件的条件。请注意,本软件的作者是 INRIA(Institut National de Recherche en Informatique et en Automatique)和 ENPC(L'Ecole Nationale des Ponts et Chaussées),只有他们才拥有该软件的产权和全部附加权利。

2. 定义

本软件是指按照版本系列方式由 INRIA 和 ENPC 发布的 SCILAB 软件及其文档资料。

SCILAB 衍生(derived)软件定义为经过用户对本软件的整体或部分进行修改、转换或者改编的软件。

SCILAB 合成(composite)软件定义为本软件整体或部分的软件内容被链接到其他软件、应用软件包或者工具箱,而这些软件、应用软件包或者工具箱是在用户所拥有、或用户受益的软

件中使用。

3. 制定本软件协议的目的和条件

(1) INRIA 和 ENPC 授权给用户可以免费方式复制本软件的源代码以及目标代码。该授权除了要求用户在全部复制品上必须注明:"SCILAB(c)INRIA-ENPC"之外,没有其他限制。

(2) INRIA 和 ENPC 授权给用户可以自由修改本软件错误,包括为链接本软件进行的任何修改以及对本软件通用功能方面的修正。然而该授权要求用户在全部修改文件部分必须加入补丁方式的文件或用等同效果的其他手段的文件,以说明修正的内容与修改时间。

(3) INRIA 和 ENPC 授权给用户可以免费方式分发本软件的源代码、目标代码以及条款(2)所述的 SCILAB 修改软件。然而该授权要求以下几点:

• 用户在分发软件时必须注明:"SCILAB (c) INRIA-ENPC"。

• 用户必须在满足本协议各项条文规定条件下分发本软件。

• 对 SCILAB 进行修正的补丁方式的文件或等同效果的其他手段文件应该是可以免费分发的。

4. 制定 SCILAB 衍生软件协议的目的和条件

(1) INRIA 和 ENPC 授权给用户可以免费方式对本软件进行整体或部分的复制、修改、转换或者改编源代码以及目标代码。该授权要求用户必须提供相应的修改、转换、或者改编本软件的补丁文件。该补丁文件同时要标明有关用户修改、转换或者改编本软件的基本内容与相应时间。经过用户对 SCILAB 本身进行的

修改、转换或者改编的 SCILAB 软件整体部分被称为 SCILAB 衍生软件。

(2) INRIA 和 ENPC 授权给用户可以免费方式使用根据条款(1)所述的 SCILAB 衍生软件。该授权除了要求用户在使用该 SCILAB 衍生软件时必须注明:"SCILAB(c)INRIA-ENPC"之外,没有其他限制。

(3) INRIA 和 ENPC 授权给用户可以免费方式以非商业目的地分发 SCILAB 衍生软件的源代码、目标代码。然而该授权要求以下几点:

- 用户在分发时必须以显著方式注明:"SCILAB inside (c) INRIA-ENPC"。
- 用户必须在满足本协议各项条文规定条件下分发SCILAB 衍生软件。
- SCILAB 衍生软件的接受者应该得到全部源码程序。
- SCILAB 衍生软件是在非 SCILAB 名义下分发的。

(4) 任何以商业为目的地使用或分发 SCILAB 衍生软件的用户必须事先征得 INRIA 和 ENPC 的授权同意。

5. 制定 SCILAB 合成软件协议的目的和条件

(1) INRIA 和 ENPC 授权给用户可以整体或部分地将本软件与用户其他自我拥有的软件、应用软件包或者工具箱进行链接方式的应用,以获得 SCILAB 合成软件。

(2) INRIA 和 ENPC 授权给用户可以免费方式使用本软件以及 SCILAB 合成软件的源代码以及目标代码。该授权除了要求用户在全部复制品上明确注明:"composite software using SCILAB (c)INRIA-ENPC functionality"之外,没有其他限制。

(3) INRIA 和 ENPC 授权给用户可以免费方式以非商业目的地分发 SCILAB 合成软件的源代码、目标代码。然而该授权要

求以下几点：

- 用户在分发时必须以显著方式注明："composite software using SCILAB (c)INRIA-ENPC functionality"。
- 用户必须在满足本协议各项条文规定条件下分发 SCILAB 衍生软件。
- SCILAB 衍生软件的接受者应该得到全部源码程序。
- SCILAB 衍生软件是在非 SCILAB 名义下分发的。

(4) 任何以商业为目的地使用或分发 SCILAB 合成软件的用户必须事先征得 INRIA 和 ENPC 授权同意。

6. 保证书方面的限制

除了特殊说明之外，本软件质量正如所提供的状态，既无任何明确的、也无隐含方式的质量与安全方面的保证。自行使用该软件后发生的危害结果全部由用户负责，即使是软件本身缺陷产生的所有损失，也将由用户自己承担。

7. 自动生成的同意书

当用户进入并使用本软件时，用户将自动被认为是已经接受本协议的全部权利和要求。

8. 关于相关效果

本协议是与其他合同有关的。然而，本软件用户不受任何与该合同或与第三方相关合同的约束。

9. 本协议适用法律

本协议中的一切条文和作用效果将受到法兰西共和国法律以及法兰西共和国最高法院的约束。

附录 C　SCILAB 部分函数指令表

（注：本指令表只收集了部分常用指令，有关全部指令请参照文档文件）

1. 通用指令

help	在线帮助
apropos	文档中关键词搜寻
ans	缺省变量名以及最新表达式的运算结果
clear	从内存中清除变量和函数
exit	关闭 SCILAB
quit	退出 SCILAB
save	把内存变量存入磁盘
exec	运行脚本文件
mode	文件运行中的显示格式
getversion	显示 SCILAB 版本
ieee	浮点运算溢出显示模式选择
who	列出工作内存中的变量名
edit	文件编辑器
type	变量类型
what	列出 SCILAB 基本命令
format	设置数据输出格式
chdir	改变当前工作目录
getenv	给出环境值
mkdir	创建目录
pwd	显示当前工作目录
evstr	执行表达式

2. 运算符和特殊算符

+	加
-	减
*	矩阵乘
.*	数组乘
^	矩阵乘方
.^	数组乘方
\	反斜杠或左除
/	斜杠或右除
./或.\	数组除

==	等号
~=	不等号
<	小于
>	大于
<=	小于或等于
>=	大于或等于
&,and	逻辑与
|,or	逻辑或
~,not	逻辑非
:	冒号
( )	圆括号
[ ]	方括号
{ }	花括号
.	小数点
,	逗号
;	分号
//	注释号
=	赋值符号
'	引号
'	复数转置号
.'	转置号
ans	最新表达式的运算结果
%eps	浮点误差容限，$=2^{-52}\approx 2.22\times10^{-16}$
%i	虚数单位 $=\sqrt{(-1)}$
%inf	正无穷大
%pi	圆周率，π=3.1415926535897…

3. 编程语言结构

abort	中止计算或循环
break	终止最内循环
case	同 select 一起使用
continue	将控制转交给外层的 for 或 while 循环
else	同 if 一起使用
elseif	同 if 一起使用
end	结束 for,while,if 语句
for	按规定次数重复执行语句
if	条件执行语句
otherwise	可同 switch 一起使用
pause	暂停模式
return	返回
select	多个条件分支
then	同 if 一起使用
while	不确定次数重复执行语句
eval	特定值计算
feval	函数特定值计算或多变量计算
function	函数文件头
global	定义全局变量
isglobal	检测变量是否为全局变量
error	显示错误信息
lasterror	显示最近的错误信息
sprintf	按格式把数字转换为串
warning	显示警告信息

4. 基本数学函数

acos	反余弦

acosh	反双曲余弦
acot	反余切
acoth	反双曲余切
acsc	反余割
acsch	反双曲余割
asin	反正弦
asinh	反双曲正弦
atan	反正切
atanh	反双曲正切
cos	余弦
cosh	双曲余弦
cotg	余切
coth	双曲余切
sin	正弦
sinh	双曲正弦
tan	正切
tanh	双曲正切
exp	指数
log	自然对数
log10	常用对数
log2	以 2 为底的对数
sqrt	平方根
abs	绝对值
conj	复数共轭
imag	复数虚部
real	复数实部
ceil	向上(正无穷大方向)取整
fix	向零方向取整
floor	向下(负无穷大方向)取整
round	四舍五入取整
sign	符号函数
gsort	降次排序
erf	误差函数
erfc	补误差函数
gamma	gamma 函数
interp	插值函数
interpln	线性插值函数
intsplin	样条插值函数
smooth	样条平滑函数
spline	样条函数
quarewave	方波函数
sign	符号函数
double	将整数转换为双精度浮点数

5. 基本矩阵函数和操作

eye	单位阵
zeros	全零矩阵
ones	全 1 矩阵
rand	均匀分布随机阵
genmarkov	生成随机 Markov 矩阵
linspace	线性等分向量
logspace	对数等分向量
logm	矩阵对数运算
cumprod	矩阵元素累计乘
cumsum	矩阵元素累计和
toeplitz	Toeplitz 矩阵
disp	显示矩阵和文字内容
length	确定向量的长度

size	确定矩阵的维数
diag	创建对角阵或抽取对角向量
find	找出非零元素 1 的下标
matrix	矩阵变维
rot90	矩阵逆时针旋转 90 度
sub2ind	据全下标换算出单下标
tril	抽取下三角阵
triu	抽取上三角阵
conj	共轭矩阵
companion	伴随矩阵
det	行列式的值
norm	矩阵或向量范数
nnz	矩阵中非零元素的个数
null	清空向量或矩阵中的某个元素
orth	正交基
rank	矩阵秩
trace	矩阵迹
cond	矩阵条件数
rcond	逆矩阵条件数
inv	矩阵的逆
lu	LU 分解或高斯消元法
pinv	伪逆
qr	QR 分解
givens	Givens 变换
linsolve	求解线性方程
lyap	Lyapunov 方程
hess	Hessenberg 矩阵
poly	特征多项式
schur	Schur 分解
expm	矩阵指数
expm1	矩阵指数的 Pade 逼近
expm2	用泰勒级数求矩阵指数
expm3	通过特征值和特征向量求矩阵指数
funm	计算一般矩阵函数
logm	矩阵对数
sqrtm	矩阵平方根

6. 特性值与奇异值

spec	矩阵特征值
gspec	矩阵束特征值
bdiag	块矩阵,广义特征向量
eigenmarkov	正则化 Markov 特征向量
pbig	特征空间投影
svd	奇异值分解
sva	奇异值分解近似

7. 矩阵元素运算

cumprod	元素累计积
cumsum	元素累计和
hist	统计频数直方图
max	最大值
mean	平均值
median	中值
min	最小值
prod	元素积
sort	由大到小排序
std	标准差

sum	元素和
trapz	梯形数值积分
corr	求相关系数或方差

8. 稀疏矩阵运算

sparse	稀疏矩阵(只存储非零元素)
adj2sp	邻接矩阵转换为稀疏矩阵
full	稀疏矩阵转换为全矩阵
mtlb_sparse	将 SCILAB 稀疏矩阵转换为 MATLAB 稀疏矩阵格式
sp2adj	将稀疏矩阵转换为邻接矩阵
speye	稀疏矩阵方式单位阵
sprand	稀疏矩阵方式随机矩阵
spzeros	稀疏矩阵方式全零阵
lufact	稀疏矩阵 LU 分解
lusolve	稀疏矩阵方程求解
spchol	稀疏矩阵 Cholesky 分解

9. 输入输出函数

diary	生成屏幕文本记录
disp	变量显示
file	文件管理
input	用户键盘输入
load	读已存的变量
mclose	关闭文件
mget	读二进制文件
mgetl	按行读 ASCII 码文件
mgetstr	读字符串中单个字
mopen	打开文件
mput	写二进制文件
mfscanf	读 ASCII 码文件
print	将变量记录为文件
read	读矩阵变量
save	将变量存为二进制文件
strartup	启动文件
write	按格式存文件
xgetfile	对话方式获取文件路径
x_dialog	建立 Xwindows 参数输入对话框
Tk_Getvar	得到 Tk 文件变量
Tk_EvalFile	执行 Tk 文件

10. 函数与函数库操作

deff	在线定义函数
edit	函数编辑器
function	打开函数定义
functions	SCILAB 函数或对象
genlib	在给定目录下建立所有文件的函数库
get_function_path	读函数库的文件存储目录路径
getd	读函数库中的全部文件
getf	在文件中定义一个函数
lib	函数库定义
macro	SCILAB 函数或对象
macrovar	输入变量个数

newfun	输出变量个数

11. 字符串操作

code2str	将 SCILAB 数码转换为字符串
convstr	字母大小转换
emptystr	清空字符串
grep	搜寻相同字符串
part	字符提取
str2code	将字符串转换为 SCILAB 数码
string	字符串转换
strings	SCILAB 对象,字符串
strcat	连接字符
strindex	字符串的字符位置搜寻
strsubst	字符串中的字符替换

12. 日期与时间

date	日期
getdate	读日期与时间
timer	CPU 时间计时

13. 二维图形函数

plot2d	直角坐标下线性刻度曲线
champ	二维向量场
champ1	由颜色箭头表示的二维向量场
contour2d	等高线图
errbar	曲线上增加误差范围框线条
grayplot	应用颜色表示的表面
xgrid	画坐标网格线
histplot	统计频数直方图
Matplot	散点图阵列

14. 三维图形函数

plot3d	三维表面
plot3d1	用颜色或灰度表示的三维表面
param3d	三维中单曲线
param3d1	三维中多曲线
contour	三维表面上的等高线图
hist3d	三维表示的统计频数直方图
geom3d	三维向二维上的投影

15. 线条类图形

xpoly	单线条或单多边形
xpolys	多线条或多多边形
xrpoly	正多边形
xsegs	非连接线段
xfpoly	单个多边形内填充
xfpolys	多个多边形内填充
xrect	矩形
xfrect	单个矩形内填充
xrects	多个矩形内填充
xarc	单个弧线段或弧圆
xarcs	多个弧线段或弧圆
xfarc	单个弧线段或弧圆填充

xfarcs	多个弧线段或弧圆填充
xarrows	多箭头

16. 图形注释,变换

xstring	图形中字符
xstringb	框内字符
xtitle	图形标题
xaxis	轴名标注
plotframe	图形加框并画坐标网格线
isoview	等尺寸比例显示(原图形窗口不改变)
square	等尺寸比例显示(原图形窗口改变)
xsetech	设置小窗口
xchange	转换实数为图形象素坐标值
subplot	设置多个子窗口

17. 图形颜色及图形文字

colormap	应用颜色图
getcolor	交互式选择颜色图
addcolor	增加新色于颜色图
graycolormap	线性灰度图
hotcolormap	热色(红到黄色)颜色图
xset	图形显示方式设定
xget	读当前图形显示方式设定
getsymbol	交互式选择符号和尺寸

18. 图形文件及图形文字

xsave	将图形存储为文件
xload	从磁盘中读出图形文件
xbasimp	将图形按 PS 文件打印或存储为文件
xs2fig	将图形生成 Xfig 格式文件
xbasc	取消图形窗及其相关内容
xclear	清空图形窗
driver	选择图形驱动器
xinit	图形驱动器初始化
xend	关闭图形
xbasr	图形刷新
replot	更改显示范围后的图形刷新
xdel	关闭图形
xname	改变当前图形窗名称

19. 控制分析用图形

bode	伯德图坐标
gainplot	幅值图坐标(伯德图中的幅值图)
nyquist	奈奎斯特图
m_circle	M-圆图
chart	尼库拉斯图
black	Black-图
evans	根轨迹图

sgrid	s 平面图
plzr	零-极点图
zgrid	z 平面图

20. 图形应用中的其他指令

graphics	图形库指令表
xclick	等待鼠标在图形上的单击输入
locate	由单击读入图形中的多点位置坐标
xgetmouse	由单击读入图形中的当前点位置坐标

21. 系统与控制

abcd	状态空间矩阵
cont_mat	可控矩阵
csim	线性系统时域响应
dsimul	状态空间的离散时域响应
feedback	反馈操作符
flts	时域响应(离散、采样系统)
frep2tf	基于传递函数的频域响应
freq	频域响应
g_margin	幅值裕量
imrep2ss	基于状态空间的脉冲响应
lin	线性化操作
lqe	Kalman 滤波器
lqg	LQG 补偿器
lqr	LQ 补偿器
ltitr	基于状态空间的离散时域响应
obscont	基于观测器的控制器
observer	观测器
obsv_mat	观测矩阵
p_margin	相位裕量
phasemag	相位与幅值计算
ppol	极点配置
repfreq	频域响应
ricc	Riccati 方程
rtitr	基于传递函数的离散时域响应
sm2ss	系统矩阵到状态空间的变换
ss2ss	反馈连接的状态空间到状态空间的变换
ss2tf	状态空间到传递函数的变换
stabil	稳定性计算
tf2ss	传递函数到状态空间的变换
time_id	SISO 系统最小方差辨识

22. 鲁棒控制

augment	被控对象增广操作
bstap	Hankel 矩阵近似

ccontrg	H_∞ 控制器
dhnorm	离散 H_∞ 范数
h2norm	H_2 范数
h_cl	闭环矩阵
h_inf	H_∞ 控制器
h_norm	H_∞ 范数
hankelsv	Hankel 矩阵奇异值
leqr	H_∞ 控制器的 LQ 增益
linf	无穷范数
riccati	Riccati 矩阵
sensi	敏感函数

23. 动态系统

arma	ARMA 模型
arma2p	基于 AR 模型获得多项式矩阵
armac	ARMAX 辨识
arsimul	ARMAX 系统仿真
noisegen	噪声信号发生器
odedi	常微分方程仿真检测
prbs_a	伪随机二进制序列发生器
reglin	线性拟合

24. 系统与控制实例

artest	Arnold 动态系统
bifish	鱼群人口发展的离散时域模型
boucle	具有观测器的动态系统相位图
chaintest	生物链模型
gpech	渔业模型
fusee	登陆火箭问题
lotest	Lorennz 吸引子
mine	采矿问题
obscontl	可控可观系统
portr3d	三维相位图
portrait	二维相位图
recur	双线性回归方程
systems	动态系统
tangent	动态系统的线性化
tadinit	动态系统的交互初始化

25. 非线性工具(优化与仿真)

bvode	边界值问题的常微分方程
dasrt	隐式微分方程过零解
dassl	代数微分方程
datafit	基于测量数据的参数辨识
derivative	导数计算
fsolve	非线性函数过零解
impl	线性微分方程
int2d	二维定积分
int3d	三维定积分
intg	不定积分
leastsq	非线性最小二乘法
linpro	线性规划
lmisolver	线性不等矩阵

ode	常微分方程
ode_discrete	离散常微分方程
ode_root	常微分方程根解
odedc	连续离散常微分方程
optim	非线性优化
quapro	线性二次型规划
semidef	半正定规划

26. 多项式计算

coeff	多项式系数
coffg	多项式矩阵逆
degree	多项式阶数
denom	分母项
derivat	有理矩阵求导
determ	矩阵行列式值
factors	因式分解
hermit	Hermit 型
horner	多项式计算
invr	有理矩阵逆
lcm	最小公倍数
ldiv	多项式矩阵长除
numer	分子项
pdiv	多项式矩阵除
pol2des	将多项式矩阵变换为表达式
pol2str	将多项式变换为字符串
polfact	最小因式
residu	余量
roots	多项式的根
simp	多项式的简化
systmat	系统矩阵

27. 信号处理

%asn	椭圆积分
%k	Jacobi 完全椭圆积分
%sn	Jacobi 椭圆函数
analpf	模拟量低通滤波器
buttmag	Butterworth 滤波器响应
cepstrum	倒谱计算
cheb1mag	Chebyshev 一型响应
cheb2mag	Chebyshev 二型响应
chepol	Chebyshev 多项式
convol	卷积
corr	相关,协方差
cspect	谱估计(应用相关法)
dft	离散傅里叶变换
fft	快速傅里叶变换
filter	滤波器建模
fsfirlin	FIR 滤波器设计
hank	协方差矩阵到 Hankel 矩阵变换
hilb	Hilbert 变换
iir	IIR 数字滤波器
intdec	信号采样率更改
kalm	Kalman 滤波器更新
mese	最大熵谱估计
mfft	多维快速傅里叶变换
mrfit	频率响应拟合
phc	Markov 过程
srkf	Kalman 滤波器平方根

sskf	稳态 Kalman 滤波器
system	观测更新
wfir	线性相位 FIR 滤波器
weiener	Weiener(维纳)滤波器
window	对称窗函数
yulewalk	最小二乘滤波器
zpbutt	Buthererworth 模拟滤波器
zpch1	Chebyshev 模拟滤波器

28. 音频信号

analyze	音频信号频域图
auread	读 *.au 音频文件
auwrite	写 *.au 音频文件
lin2mu	将线性信号转换为 μ 率码信号
loadwave	取 *.wav 音频文件
mapsound	音频信号图示
mu2lin	将 μ 率码信号转换为线性信号
playsnd	音频信号播放
savewave	存 *.wav 音频文件
wavread	读 *.wav 音频文件
wavwrite	写 *.wav 音频文件

29. 语言与数据转换工具

ascii	字符串的 ASCII 码
excel2sci	读 ASCII 格式的 Excel 文件
fun2string	将 SCILAB 函数生成 ASCII 码
mfile2sci	将 MATLAB 的 M 格式文件转换为 SCI 格式文件
mtlb_load	取 MATLAB 第 4 版本文件中变量
mtlb_save	按 MATLAB 第 4 版本文件格式存变量
pol2tex	将多项式转换为 TeX 格式
sci2for	将 SCILAB 函数转换为 FORTRAN 格式文件
texprint	按 TeX 格式输出 SCILAB 对象
translatepaths	将子目录下的所有 MATLAB 文件转换为 SCI 文件格式